VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Konfigurační modul univerzální automobilové sběrnicové brány

Configuration module of universal car gateway

Bc. Jan Jašek

Diplomová práce

Vedoucí práce: doc. Ing. Petr Bilík, Ph.D.

Ostrava, 2021

Abstrakt

Cílem této práce je realizovat konfigurační modul sběrnicové brány, který umožní její plnou konfiguraci prostřednictvím webové aplikace běžící na vestavěném webovém serveru, realizovaném na HW platformě ESP32. A to z téměř jakéhokoli zařízení obsahujícího rozhraní WiFi a internetový prohlížeč bez ohledu na použitý operační systém či velikost a rozlišení obrazovky. Diplomová práce je vypracována ve spolupráci s firmou Porsche Engineering Services, s.r.o. Úvodní část této práce je věnována teorii sběrnicových bran a také nejrozšířenějším automobilovým sběrnicím CAN, LIN a FlexRay. Teoretická část dále obsahuje informace o webových serverech z pohledu hardware i software a jejich využití ve vestavěných systémech. Praktická část této práce je pak věnována výběru vhodné hardwarové platformy pro realizaci a výběru vývojového prostředí pro tuto platformu. Dále následuje systémová architektura, hardwarová architektura, návrh jednotlivých funkcionalit sběrnicové brány, návrh a implementace responzivní webové aplikace, vytvoření WiFi přístupového bodu, přístup k souborům pro vytvoření webu uložených na SD kartě a návrh a implementace webového serveru a REST API. Další bod praktické části tvoří implementace komunikace mezi konfigurační a sběrnicovou částí zařízení. Závěrečná část této práce je věnována ověření funkčnosti a zhodnocení.

Klíčová slova

CAN, webový server, sběrnicová brána, ESP32

Abstract

The main goal of this thesis is create configuration modul of bus gateway for configuration bus gateway by web application running on embedded web server. Web server is realized on HW platform ESP32. Configuration of bus gateway should be possible from almost any device regardless of screen size or operating system. Wifi interface and web browser is required. The diploma thesis is realized in cooperation with Porsche Engineering Services, s.r.o. The first part of this thesis is focused on theory of bus gateways and commonly used automotive buses (CAN, LIN, FlexRay). Theoretical part also contain information about HW and SW of web servers and usage of web servers in embedded systems. The practical part of this thesis contains choosing the convenient hardware platform and convenient software environment. The practical part also contains hardware architecture, software architecture, design of gateway functionalities, design and implementation of responsive web application, implementation of WiFi access point, access to web application files saved on SD card and design and implementation of web server and REST API. Next section of practicle part is implementation of communication between configuration and control part of gateway. Final part of this thesis is dedicated to functionality verification and final evaluation.

Keywords

CAN, web server, bus gateway, ESP32

Poděkování

Rád bych na tomto místě poděkoval Ing. Martinu Holainovi za cenné rady a odbornou pomoc při realizaci této práce. Také bych rád poděkoval doc. Ing. Petru Bilíkovi, Ph.D za vedení této práce. V neposlední řadě děkuji své rodině za podporu během celého studia.

Obsah

Seznam použitých zkratek a symbolů

AP	–	Access Point
API	–	Application Programming Interface
CAN	–	Controller Area Network
CSS	–	Cascading Style Sheets
DLC	–	Data Length Code
ECU	–	Electronic Control Unit
FAT	–	File Allocation Table
FTDMA	–	Flexible Time Division Multiple Access
HTML	–	Hypertext Markup Language
HTTP	–	HyperText Transfer Protocol
IDE	–	Integrated Development Environment
IDF	–	IoT Development Framework
JS	–	JavaScript
JSON	–	JavaScript Object Notation
LIN	–	Local Interconnect Network
LXI	–	LAN eXtensions for Instrumentation
REST	–	Representational State Transfer
SOAP	–	Simple Object Access Protocol
SPI	–	Serial Peripheral Interface
TDMA	–	Time Division Multiple Access
UPS	–	Uninterruptible Power Supply
URL	–	Uniform Resource Locator
URI	–	Uniform Resource Identifier
USB	–	Universal Serial Bus
VFS	–	Virtual File System
WPA	–	Wi-Fi Protected Access
XHR	–	XMLHttpRequest

Seznam obrázků

Kapitola 1

Úvod

V automobilovém průmyslu se v současné době využívá řada komunikačních sběrnic. Mezi nejrozšířenější sběrnice patří CAN, LIN a FlexRay. Sběrnicové brány jsou zařízení sloužící nejčastěji k propojení různých typů sběrnic, ale také pro propojení sběrnic stejného typu a manipulaci se zprávami na těchto sběrnicích. Na trhu existuje množství CAN sběrnicových bran od renomovaných výrobců, které jsou velmi dobře zpracované a ověřené. Avšak u těchto sběrnicových bran je hardware a software dále nemodifikovatelný a ne všechny je možno připojit bezdrátově k zařízení sloužícímu k nastavení těchto sběrnicových bran. Zde začíná podstata této diplomové práce, která spočívá v realizaci konfigurační části sběrnicové brány tak, aby bylo k této sběrnicové bráně možno přistupovat vzdáleně, prostřednictvím téměř jakéhokoli chytrého zařízení bez ohledu na jeho operační systém či velikost a rozlišení obrazovky a takto modifikovat veškeré parametry této sběrnicové brány. Výlučným požadavkem na zařízení ze kterého může být sběrnicová brána ovládána je vybavení WiFi rozhraním a internetový prohlížeč.

Teoretická část této práce je věnována automobilovým sběrnicovým branám, základním automobilovým sběrnicím, jejich vlastnostem, parametrům a použití. Dále se v této části zabývám webovými servery, jejich vlastnostmi, využitím, hardwarovou a softwarovou částí těchto serverů, základními úkoly webového serveru a rozdíly mezi statickými a dynamickými servery. V neposlední řadě je zde věnována část i využití webových serverů ve vestavěných zařízeních.

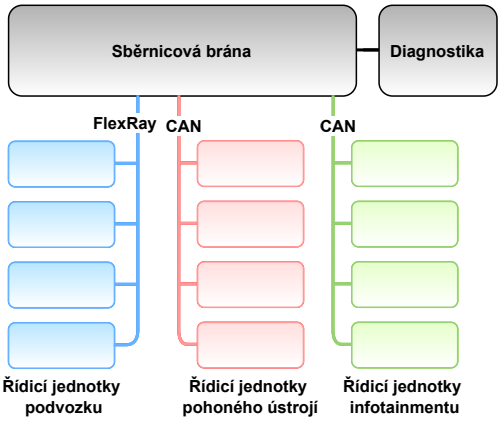
Praktická část obsahuje výběr vhodné hardwarové platformy pro realizaci této práce a výběr vhodného vývojového prostředí pro tuto hardwarovou platformu. Dále následuje zpracování hardwarové a systémové architektury, návrh jednotlivých funkcionalit sběrnicové brány, návrh a implementace webové aplikace s využitím Bootstrap front-end frameworku. Tato webová aplikace zahrnuje přihlašovací obrazovku sběrnicové brány a hlavní stránku sběrnicové brány pomocí které je realizováno samotné nastavení parametrů sběrnicové brány uživatelem. Praktická část dále obsahuje realizaci WiFi přístupového bodu, vyčítání souborů webu z SD karty s ošetřením situace kdy SD karta není vložena, nebo potřebná data na ni nejsou dostupná, pomocí integrovaného webu ve vnitřní paměti zařízení. Následuje návrh a implementace webového serveru a REST API pro komunikaci mezi

webovou aplikací a webovým serverem. Další bod praktické části obsahuje implementaci komunikace mezi konfigurační a sběrnicovou částí sběrnicové brány. Poslední dva body praktické části této práce jsou věnovány ověření funkčnosti a celkovému zhodnocení této práce.

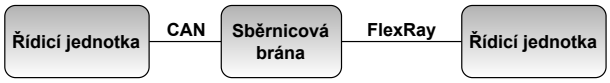
Kapitola 2

Automobilové sběrnicové brány

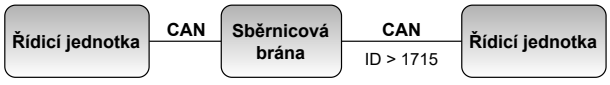
Sběrnicové brány jsou často využívaná zařízení pro testování a vývoj v oblasti automotive. Jedná se o zařízení, která mohou sloužit jako rozbočovače signálu, dále pro sledování signálů na sběrnici, pro manipulaci s daty nebo k úpravě dat pro propojení různých typů sběrnic viz (Obrázek 2.1). Vozidla v dnešní době obsahují velké množství řídicích jednotek (ECU), které umožňují implementovat do vozidel složitější a pokročilejší funkce. Sběrnicová brána může sloužit jako prostředník mezi různými sběrnicemi, kterými jsou tyto řídicí jednotky propojeny. Může se tedy jednat o prostředníka pro výměnu dat mezi sběrnicemi jako je CAN, LIN a FlexRay viz (Obrázek 2.2a) nebo mezi stejnými sběrnicemi s jinými přenosovými rychlostmi. Stejně tak se však může jednat například pouze o CAN sběrnicovou bránu, která neslouží jako prostředník mezi různými sběrnicemi, ale slouží pro manipulaci s daty mezi jednotkami propojenými pomoci totožné sběrnice viz (Obrázek 2.2b), kdy může mít sběrnicová brána funkcionalitu filtru, nebo může jiným způsobem manipulovat se zprávami. Různé sběrnicové brány mohou mít různě široký rozsah hardwarových a softwarových možností a mohou se od sebe velmi lišit. Mezi základní parametry patří počet podporovaných typů sběrnic, počet fyzických konektorů pro připojení těchto sběrnic a způsob konfigurace. [6]



Obrázek 2.1: Architektura sběrnicové brány



(a) Použití sběrnicové brány jako interface mezi rozdílnými typy sběrnic.



(b) Použití sběrnicové brány pro filtrování dat dle priority.

Obrázek 2.2: Možnosti použití sběrnicové brány

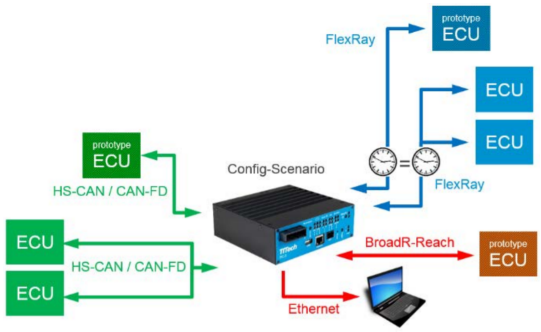
2.1 Dostupná řešení sběrnicových bran na trhu

Na trhu existuje řada sběrnicových bran od renomovaných, ale i méně známých výrobců viz kapitoly 2.2 a 2.3. Nabízené sběrnicové brány se od sebe vzájemně velmi liší množstvím podporovaných typů sběrnic, maximálním počtem sběrnic které mohou být současně připojeny a způsobem konfigurace. Ze zmíněných sběrnicových bran produkt TTTech PC-3 umožňuje výměnu dat mezi sběrnicemi CAN, CANFD, FlexRay a automotive Ethernet. Oproti tomu produkt RACELOGIC CAN Gateway je striktně zaměřen na sběrnici CAN a podporuje pouze CAN a CANFD. Sběrnicová brána od RACELOGIC je navržena pro použití přímo v automobilu po čas jízdy pro sledování toku dat na sběrnici. Sběrnicová brána od značky TTTech je naopak řešena a designována spíše pro prototypování a laboratorní použití (avšak může být použita i přímo ve vozidlech v provozu). Ani jedna z výše zmíněných sběrnicových bran však nedisponuje možností vzdálené konfigurace prostřednictvím jakéhokoli mobilního zařízení či notebooku. Výše zmíněné sběrnicové brány musí být konfigurovány z PC pomocí USB nebo Ethernetu. Návrh řešení tohoto inovativního přístupu je hlavní částí této diplomové práce.

2.2 TTTech PC-3

TTTech je společnost, která se zabývá vývojem zařízení pro použití v automobilovém, železničním, ale také leteckém průmyslu. V nabídce této společnosti v oblasti automobilového průmyslu lze nalézt univerzální automobilovou sběrnicovou bránu PC-3 viz (Obrázek 2.3), která umožňuje výměnu dat mezi sběrnicemi CAN, CANFD, FlexRay a automotive Ethernet. Dále tato sběrnicová brána podporuje manipulaci s daty v reálném čase, což umožňuje provádět analýzy sběrnic a systémů ve fázích vývoje, integrace a testování. Vybraná data mohou být předávána do dataloggeru pro analýzu nebo na jiné zobrazovací zařízení viz (Obrázek 2.4). Konfiguraci zařízení lze provádět prostřednictvím software Slate XNS pomocí drag-and-drop grafického rozhraní. [5]

Obrázek 2.3: Univerzální sběrnicová brána od společnosti TTTech. [5]

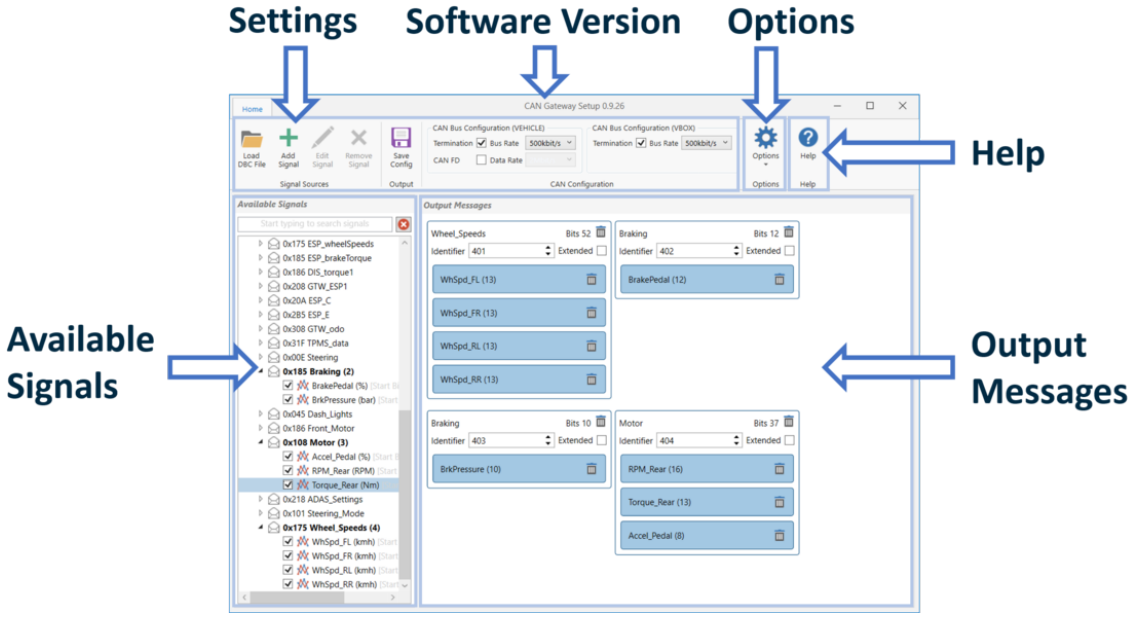


Obrázek 2.4: Architektura univerzální sběrnicové brány společnosti TTTech. [5]

2.3 RACELOGIC CAN Gateway

Společnost RACELOGIC se zabývá návrhem a výrobou elektronických systémů pro měření, záznam, zobrazení, analýzu a simulaci dat z vozidel. V nabídce této společnosti lze nalézt zařízení pro přesné měření zrychlení a zpomalení, zařízení pro velmi přesné získávání dat z vozidel na závodních okruzích a mimo jiné tato společnost nabízí i CAN Gateway viz (Obrázek 2.5). Jedná se o sběrnicovou bránu sloužící k řízení toku dat mezi dvěma samostanými sběrnicemi. Umožňuje převést zprávy CANFD na CAN, izolovat 2 sběrnice CAN ale zároveň umožnit přenos všech zpráv mezi nimi, převést data z CAN sběrnice o určité rychlosti (baudrate) na sběrnici CAN s jinou rychlostí, filtrovat data z jedné sběrnice CAN a předat konkrétní data na jinou sběrnici CAN. Nastavení tohoto zařízení je možno provádět prostřednictvím USB kabelu z počítače pomocí poskytovaného software viz (Obrázek 2.6). V kombinaci s dalšími zařízeními z portfolia této společnosti je možné provádět loggování dat a tyto data zobrazovat, uchovávat a dále zpracovávat. [3]

Obrázek 2.5: CAN sběrnicová brána od společnosti RACELOGIC. [3]

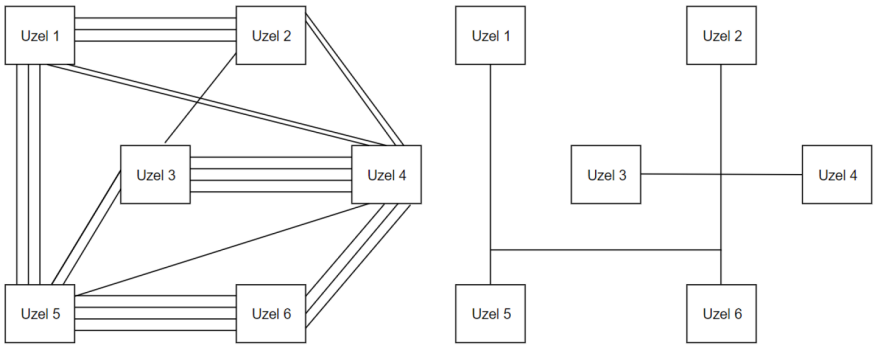


Obrázek 2.6: SW pro nastavení sběrnicové brány společnosti RACELOGIC. [4]

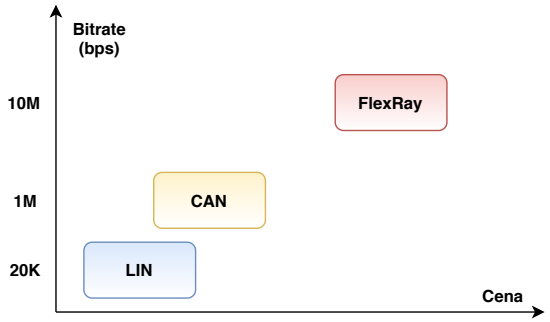
Kapitola 3

Automobilové sběrnice

Většina dnešních vozidel je již vybavena velkým množstvím elektronických řídicích systémů. Tato situace je způsobena stále se zvyšujícími požadavky EU na emisní limity, snižování spotřeby paliva, ale taktéž vysokými nároky uživatelů na komfort. Nutnost interakce mezi funkcemi které jsou v těchto systémech implementovány byla důvodem realizace vzájemné komunikace mezi všemi těmito systémy. V běžných systémech jsou signály přenášeny po vlastních přenosových linkách, což by v případě automobilů bylo z důvodu obrovského množství přenášených signálů finančně i realizačně velmi náročné. Proto jsou všechny jednotky v automobilu u kterých je požadováno aby komunikovaly mezi sebou nebo se senzory zajišťujícími sběr informací, vzájemně propojeny pomocí sběrnic viz (Obrázek 3.1). U osobních vozidel jsou využívány především sběrnice CAN, LIN a ve stále větší míře i FlexRay. Výběr konkrétní sběrnice se odvíjí od řady požadavků jako například finanční náklady a rychlost komunikace viz (Obrázek 3.2) [2].



Obrázek 3.1: Porovnání propojení ECUs pomocí samostatných přenosových linek a pomocí sběrnic



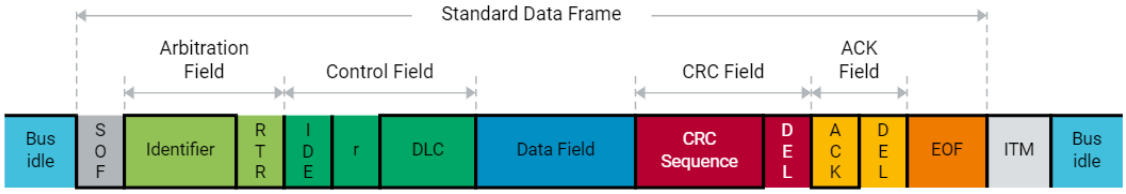
Obrázek 3.2: Srovnání sběrnic v závislosti rychlosti na ceně

3.1 CAN

CAN (Controller Area Network) je sériovou datovou sběrnicí po které probíhá komunikace pomocí zpráv. Sběrnice CAN byla vyvinuta firmou Robert Bosch v roce 1986 pro použití v automobilovém průmyslu ke komunikaci více mikrokontrolerů mezi sebou. Hlavní výhodou CANu, která jej odlišuje od ostatních komunikačních protokolů je využití broadcasového vysílání. Nejedná se tedy o komunikaci Master-Slave, ale aktivita na sběrnici je monitorována všemi uzly v síti a každý uzel může být v pozici Master. Není tedy možné odeslat konkrétní zprávu z uzlu A pouze do uzlu B. Tato zpráva je přijata všemi uzly, které jsou součástí sítě a uzly pro které tato zpráva není určena (to zjistí příjemce podle informací z identifikátoru) ji jednoduše zahodí. Každá zpráva obsahuje identifikátor, který zároveň slouží pro určení priority jejího vysílání na sběrnici. Přístup k přenosovému médiu je řízen pomocí sběrnice s náhodným přístupem. Tato sběrnice řeší kolize na základě prioritního rozhodování (velikosti priority obsažené v identifikátoru). Ve chvíli kdy se více uzlů v síti pokouší o přístup ke sběrnici současně, dochází k arbitráži. Při arbitráži jsou porovnávány identifikátory zpráv. Čím nižší je hodnota identifikátoru, tím větší má zpráva prioritu. Zprávy s vysokou prioritou získají přístup na sběrnici dříve. Zprávy s nízkou prioritou, které arbitráž prohrály, zahájí novou arbitráž v okamžiku kdy je sběrnice opět k dispozici. Jde tedy o distribuované řízení, které není přímo závislé na jednom jediném nadřazeném uzlu, který by řídil celou síť. Při vyřazení jednoho z uzlů ze sítě může zbytek fungovat dál. Toto je jeden ze základních důvodů vysoké spolehlivosti a možnosti tzv. plug and play připojování a odpojování uzlů. CAN je sériový, half-duplex, asynchronní komunikační protokol. Což znamená, že přenos je realizován po jednotlivých bitech (sekvenčně). Zprávy mohou putovat oběma směry, ale ne v jeden okamžik. V případě dvou uzlů tedy vždy jeden vysílá a druhý přijímá. Nikdy však nemohou vysílat oba současně. CAN protokol je asynchronní a není zde potřeba žádný synchronizační hodinový signál mezi vysílačem a přijímačem. Pro přenos je použita kroucená dvoulinka. Díky svým vlastnostem a parametrům jako je vysoká spolehlivost a zabezpečení proti chybám, vysoká přenosová rychlost, snadné nasazení a rozšiřitelnost se tento komunikační protokol rychle rozšířil z oblasti automobilového průmyslu i do dalších oblastí, kde je kladen důraz na bezpečnost a nutnost komunikace v reálném čase, jako jsou například průmyslová

automatizace, výtahy, lékařské přístroje atd. Sběrnice CAN je složena z CAN protokolu a samotné fyzické vrstvy. Low speed CAN (125 kbps) je využíván na místech kde není vyžadována maximální rychlost (lepší odolnost proti poruchám). High speed CAN (1 Mbps) je využíván v situacích, kde je požadována vyšší rychlost. S délkou vedení se maximální rychlost značně snižuje. Protokol CAN je definován normou ISO 11898 a existuje ve dvou specifikacích. Jedná se o specifikaci 2.0A a CAN 2.0B. Rozdílem mezi těmito specifikacemi je délka identifikátoru zprávy viz (Obrázek 3.3). CAN 2.0A – 11bit identifikátor (standardní) CAN 2.0B – 29bit identifikátor (prodloužený) [2] [11] [8] [7].

3.1.1 CAN Frame



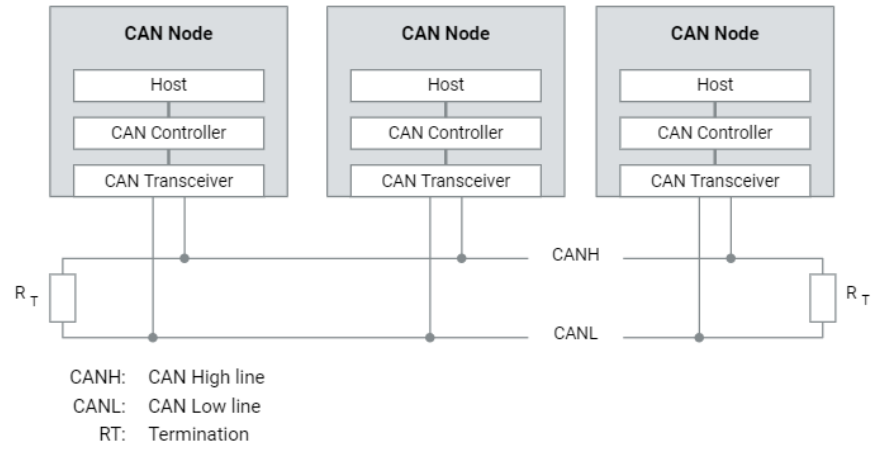
Obrázek 3.3: CAN frame [8]

- SOF – Start of frame.
- Identifier – Slouží k identifikaci a udává prioritu zprávy, která poté slouží k rozhodování, která zpráva má při přenosu na sběrnici přednost.
- RTR – Remote Transmission Request udává, zda se jedná o data Frame nebo Remote Frame. V případě kdy se jedná o Data Frame je hodnota RTR na úrovni dominant (log.0). V případě kdy se jedná o Remote Frame (request) je RTR na úrovni recessive (log.1).
- IDE – Identifier of Extension – nabývá hodnoty 0 nebo 1 a tím udává, zda se jedná o standartní nebo prodloužený identifikátor.
- DLC – Data Length Code – Udává pomocí 4 bitů velikost dat v datovém poli (0 – 8 byte).
- Data Field – zde jsou uložená data.
- CRC Sequence – Cyclic redundancy check je kontrolní součet, který slouží ke kontrole, zda se data při přenosu neporušila. Odesílatel spočítá CRC před odesláním Frame, po doručení příjemce také spočítá CRC a následně je porovná. Pokud se nerovnají, příjemce vyšle Error Frame.
- ACK – Recievers Acknowledgement.
- EOF – End of frame.

[8]

3.1.2 Fyzická vrstva CAN

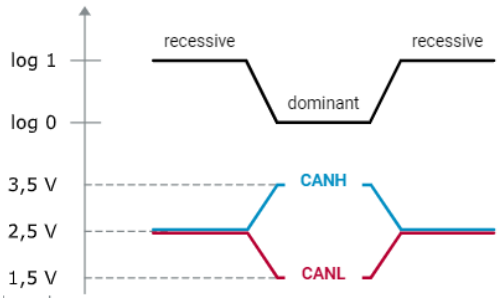
Typická struktura sběrnice CAN s připojenými zařízeními viz (Obrázek 3.4).



Obrázek 3.4: Fyzická vrstva CAN [10]

- HOST - Jako Host viz (Obrázek 3.4) je zde označen mikrokontroler, který vykonává nějaké své specifické úkoly a rozhoduje co znamená přijatá zpráva nebo zpráva, kterou chce odeslat.
- CAN Controller – CAN Controller viz (Obrázek 3.4) pracuje s komunikačními funkcemi a vyvolává přerušení při přenosu nebo příjmu CAN zpráv.
- CAN Transceiver – CAN Transceiver viz (Obrázek 3.4) je zodpovědný za odesílání a příjem dat po CAN sběrnici. Konvertuje elektrický signál získaný z CAN sběrnice na data, která je schopen CAN Controller zpracovat.
- K přenosu dat je využíváno diferenciálního napětí mezi CANH a CANL. Kroucená dvoulinka je využita z důvodu vysoké odolnosti vůči rušení.
- Odpory RT na obou koncích vedení mají hodnotu 120 Ω

V terminologii CAN je logická 1 nazvaná jako recesivní (recessive) a logická 0 nazvaná jako dominantní (dominant). Je zde využíváno diferenciálního napětí mezi CANH a CANL. Pokud je napětí CANH i CANL na hodnotě 2,5V jedná se o recesivní stav (log.1) a sběrnice je nečinná. V případě kdy je na CANH zvýšeno napětí na hodnotu 3,5 V a zároveň na CANL sníženo na 1,5 V je mezi nimi rozdíl 2 V a jedná se o dominantní stav (log.0). Pokud je sběrnice v dominantním stavu (log.0) což je způsobeno některým z uzlů. Nelze změnit stav sběrnice na recesivní (log.1) jiným uzlem. Sběrnice je v recesivním stavu pouze v případě kdy jsou všechny uzly ve stavu log. 1 (recesivní) viz (Obrázek 3.5) [2] [11].



Obrázek 3.5: Signálové úrovně CAN [9]

3.2 LIN

LIN (Local Interconnect Network) je sériovou jednovodičovou asynchronní sběrnicí. Tato sběrnice byla navržena pro využití v automobilovém průmyslu. Jedná se o sběrnici, která je používaná pro účely jako ovládání klimatizace, polohování sedadel a podobně, kde by bylo použití sběrnice CAN ve srovnání se sběrnicí LIN zbytečně drahé. LIN sběrnice disponuje nejmenší přenosovou rychlostí ze zde zmíněných sběrnic. LIN je single-master/ multiple-slave sběrnicí. Jedno zařízení je zde tedy jako master a má kontrolu nad ostatními zařízeními, označenými jako slave. U této sběnice nemusí být řešen přístup ke sběrnici jako například u sběrnice CAN, protože veškerý provoz na síti je řízen masterem. To že se jedná o jednovodičovou sběrnici má vliv i na cenu, která je ve srovnání s ostatními sběrnicemi nižší. Cenu dále ovlivňují nižší nároky na výkon použitého mikrokontroléru ve srovnání s CAN sběrnicí. Slave jednotky nepotřebují pro svou činnost přesné krystalové oscilátory, ale vystačí si s RC oscilátory. Maximální přenosová rychlost je 20 kbit/s. Sériový přenos dat využívá formátu UART/RS-232. [12]

3.3 FlexRay

Jedná se o vysokorychlostní sběrnici, která je využívána primárně v aplikacích které vyžadují nejvyšší možnou rychlost a vysokou bezpečnost. Přenosová rychlost této sběrnice dosahuje až 10 Mbit/s. FlexRay obsahuje dva nezávislé komunikační kanály A a B. Je možné tyto kanály používat současně a zvýšit tak přenosovou rychlost na 20 Mbit/s. Při použití pouze kanálu A slouží kanál B jako záložní pro případ selhání kanálu A. Sběnice může být založena na různých síťových topologiích jako například aktivní hvězda, kaskádová hvězda, dvoukanálová sběrnice nebo mix těchto topologií. Ke sběnici FlexRay je možno přistupovat metodou FTDMA (Flexible Time Division Multiple Access) nebo metodou TDMA (Time Dividison Multiple Access).

- TDMA - Při použití této metody je předem definován komunikační plán a každý uzel může komunikovat ve svém časovém slotu.

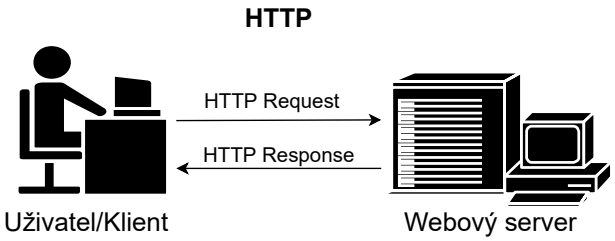
- FTDMA - Používá se v případě kdy dochází k asynchonním procesům. Zprávy nejsou tedy posílány v předem daných intervatel, ale na základě událostí.

Fyzická vrstva je u této sběrnice stejně jako u CANu založena na přenosu diferenciálních napětí. Pro přenos se používá nestíněná kroucená dvoulinka. Z důvodu eliminace možných odrazů signálu je sběrnice zakončena odpory s impedancí 80 až 100 Ω. [13] [1]

Kapitola 4

Webový server

Pod pojmem Webový server je zahrnuto spojení částí webový server jako hardwarová část a webový server jako softwarová část. Na webovém serveru (HW) je spuštěn webový server (SW). Webový server jako program (běžící na nějakém zařízení) nabízí své služby dalším zařízením. V okamžiku kdy chce klient otevřít určitou webovou stránku, vyžádá si ji prohlížeč (klient) od serveru pomocí HTTP (HyperText Transfer Protocol) viz (Obrázek 4.1). V situaci kdy prohlížeč potřebuje soubor uložený na webovém serveru, vytvoří požadavek pomocí HTTP (HTTP Request). Tento požadavek musí obsahovat URL (Uniform Resource Locator) adresu souboru. Po obdržení tohoto požadavku webový server vyhledá ve své paměti požadovaný soubor a odešle jej opět pomocí HTTP (HTTP Response) prohlížeči (klient). V případě kdy daný soubor webový server nenalezne, vrací prohlížeči chybu 404. [15] [27] [14]



Obrázek 4.1: Komunikace mezi klientem a serverem

4.1 HW Webového serveru

Webový server z pohledu hardware může být počítač nebo jiné zařízení (například vestavěné), které nabízí své služby ostatním zařízením. V globálním kontextu se však pro tyto účely využívají v případě velkých firem vlastní serverovny, v případě běžných webových stránek serverovny, které prostor na svých serverech pronajímají. Mezi hlavní problémy patří chlazení (proto jsou velké serverovny klimatizované), rychlost připojení k internetu, běžná údržba nebo případné hrozící výpadky

napájení. V případě služeb u kterých je dostupnost vyžadována 24 hodin denně, 7 dní v týdnu jako například internetové bankovnictví nebo přístup do e-mailové schránky, kde by k nedostupnosti bez předchozího varování nemělo dojít, je toto řešeno pomocí zdvojeného napájení. Každé zařízení dostává energii ze dvou nezávislých napájecích zdrojů UPS (Uninterruptible Power Supply). Každý z těchto zdrojů je schopen při výpadku druhého zdroje sám napájet celý systém.
V paměti zařízení poskytujícího webový server (flash paměť nebo paměťová karta v případě vestavěného zařízení, pevný disk v případě běžného počítače nebo serveru v serverovně) jsou uloženy jednotlivé komponenty webové stránky ve formě souborů. Jedná se o HTML (Hypertext Markup Language) dokumenty, obrázky, CSS (Cascading Style Sheets) stylizační šablony a JS (JavaScript) soubory. Tento webový server (HW) je připojen k internetu a poskytuje tato data, která jsou na něm uložena ostatním zařízením připojeným k síti. [26] [15]

4.2 SW Webového serveru

Webový server z pohledu software zahrnuje několik částí, které dohromady tvoří proces jak uživatelé webu přistupují k souborům na webovém serveru. Hlavním úkolem webového serveru je implementovat HTTP. Jedná se o textový protokol a všechny jeho části jsou tedy pro člověka čitelné. Samotný protokol neumožňuje šifrování ani zabezpečení integrity dat. Pro zabezpečenou komunikaci se používá HTTPS (Hypertext Transfer Protocol Secure). HTTP umožňuje prohlížeči vyslat požadavek na který musí server zareagovat. Ačkoli jsou dnešní weby již velmi propracované, je HTTP jednoduchý protokol. Jediné co tento protokol dělá je přenos požadavků (HTTP request) a odpovědí (HTTP response). Požadavky mohou provádět pouze klienti na servery a reagovat pouze servery na klienty. Při žádosti o soubor prostřednictvím tohoto protokolu musí klienti poskytnout URL adresu souboru. Webový server je povinen odpovědět na každý požadavek alespoň chybovou zprávou. Neexistuje zde žádná možnost rozhodování, žádné scripty ani kód, který by bylo možné provést. Taktéž zde neexistuje koncept dynamického vytváření stránek. Pokud se stránka bude vytvářet dynamicky, musí tuto činnost vykonat jiný program a uložit výsledek tam, kde jej webový server očekává viz kapitola 4.3. HTTP server rozumí URL adresám neboli webovým adresám například *https://www.fei.vsb.cz/cs/index.html* a devíti operacím, které se označují jako metody. [27] [15]

4.2.1 Základní HTTP metody

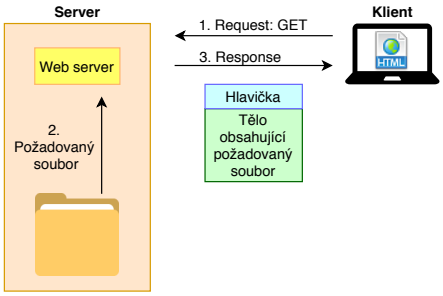
- GET - Načtení požadovaných dat identifikovaných pomocí URL adresy.
- POST - Odeslání dat na server (například odeslání vyplněného formuláře).
- HEAD - Podobná metodě GET, ale není vracen žádný obsah. Poskytovány pouze metadata o požadovaném cíli.
- PUT - Odesílání dat na specifickou URL. Pokud zde již nějaká data existují, jsou přepsána. Pokud se zde žádná data nenachází, jsou zde uložena odeslaná data.
- DELETE - Požadavek na smazání objektu identifikovaného pomocí URL ze serveru.
- OPTIONS - Dotaz jaké metody poskytuje konkrétní URL nebo celý server.
- TRACE - Odeslání kopie obdrženého požadavku zpět odesílateli, takže klient muže zjistit co na požadavku mění nebo přidávají servery, kterými požadavek prochází.
- CONNECT - Spuštění obousměrné komunikace.
- PATCH - Slouží k částečné úpravě zdroje. [16]

4.2.2 Základní stavové kódy HTTP

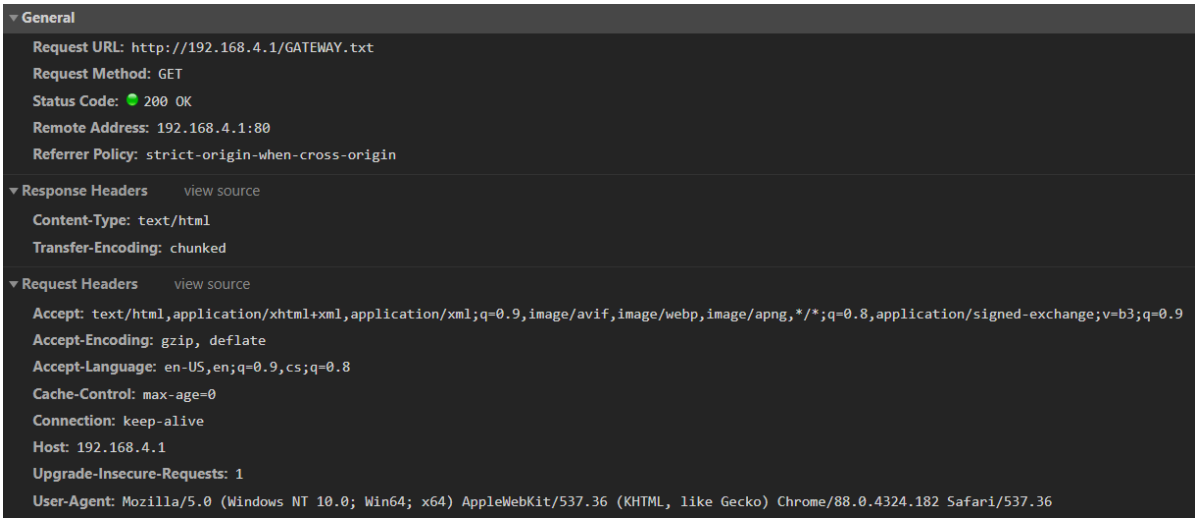
- 200 - (OK) Request byl úspěšný.
- 400 - (Bad request) Server nemůže nebo nebude zpracovávat požadavek kvůli chybě klienta (např. chybná syntaxe požadavku).
- 401 - (Unauthorized) Klient není k požadavku autorizován.
- 403 - (Forbidden) Klient nemá oprávnění.
- 404 - (Not Found) Požadavek na neexistující zdroj.
- 405 - (Method not allowed) Metoda není povolena. Např. je povolena pouze GET.
- 406 - (Not Acceptable) Server není schopen poskytnout požadovaný formát.
- 415 - (Unsupported Media Type) Server nepodporuje předaný Content-Type. [17]

4.2.3 Metoda GET

Metoda GET viz (Obrázek 4.2) slouží k získání dat umístěných na konkrétní URL. Odpověď je složena z HTTP hlavičky a následují data. Při kliknutí na odkaz ve webovém prohlížeči pro přechod na novou stránku je odeslán požadavek GET viz (Obrázek 4.3) na server s dotazem na stránku HTML umístěnou na adrese URL na kterou uživatel kliknul. [27]



Obrázek 4.2: HTTP metoda GET

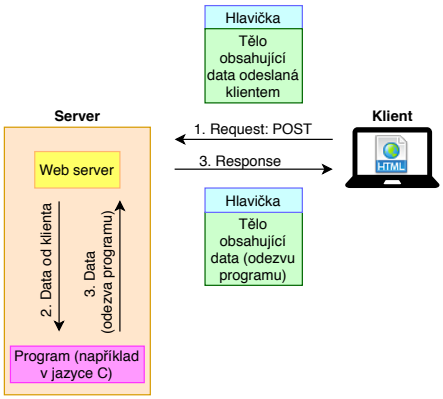


Obrázek 4.3: HTTP metoda GET data

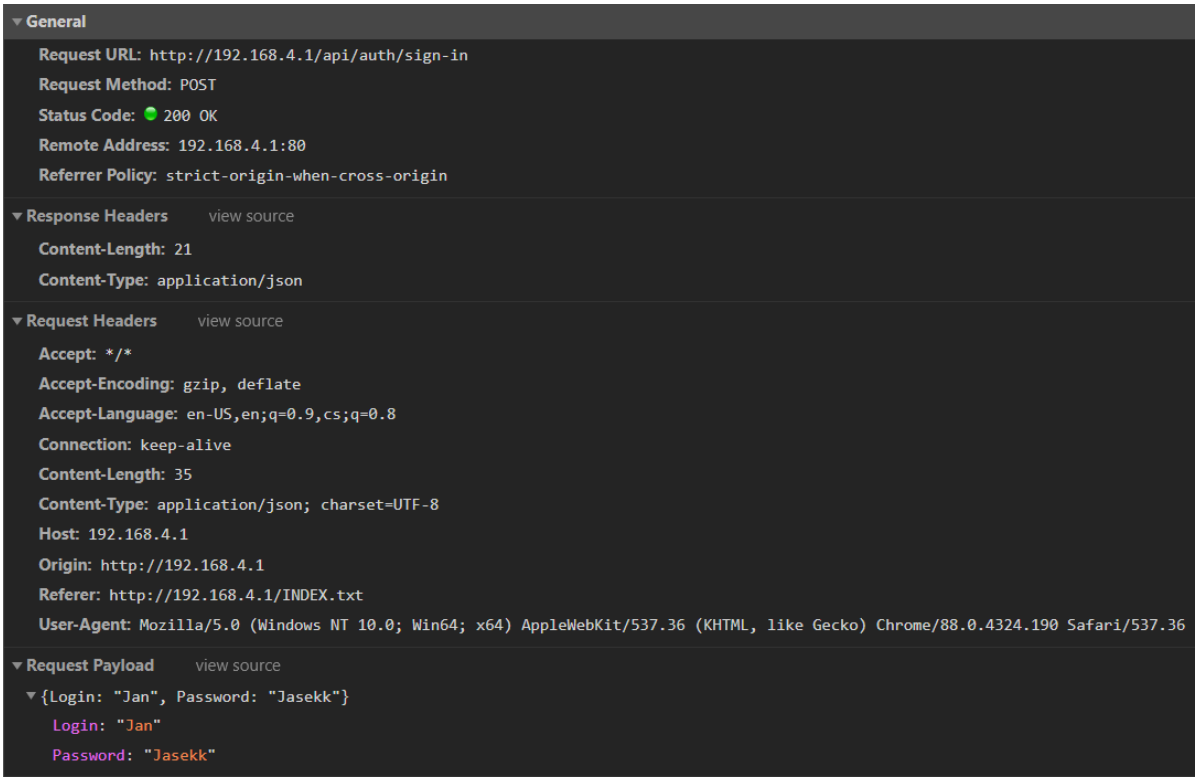
4.2.4 Metoda POST

Metoda POST se používá pro odeslání dat ze strany klienta (například webového prohlížeče) na server. Tento požadavek je využíván například pro odeslání dat z vyplněného formuláře. Základní webový server však není schopen tento požadavek zpracovat, protože nerozumí datům obsaženým v požadavku. Webový server tedy provede metodu POST (Obrázek 4.4), ale o zpracování dat obdržených v tomto požadavku se musí postarat jiný program nebo nástroj. Většina webových aplikací potřebuje mnohem rozsáhlejší funkcionalitu. Zejména v případě vestavěných systémů je

požadavkem aby webová aplikace byla schopna ovlivnit provoz zařízení. Funkcionalita webového serveru jako takového nepřesahuje schopnost jednoduchého zpracování HTTP. Z tohoto důvodu je součástí vývoje systému také nízkoúrovňový kód, který interaguje přímo s hardwarem. Tento nízkoúrovňový kód je zpravidla psán v jazyce C. Tyto programové rutiny jsou specifické pro daný systém a jsou nutností bez ohledu na to jak je implementované webové připojení. Požadavek metody POST viz (Obrázek 4.5). [27]



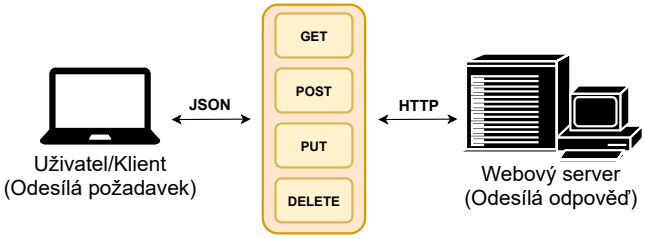
Obrázek 4.4: HTTP metoda POST



Obrázek 4.5: HTTP metoda POST data

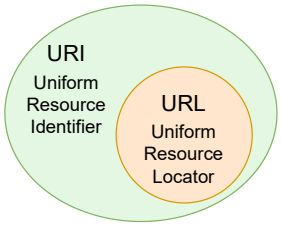
4.2.5 REST

REST (Representational State Transfer) je architekturou rozhraní pro návrh síťových aplikací, navrženou pro distribuované prostředí. Základní myšlenkou REST je namísto použití složitých mechanismů pro propojení mezi zařízeními použít jednoduchý HTTP viz (Obrázek 4.6). REST rozhraní se používá pro jednotný a snadný přístup ke zdrojům (resources). Jako zdroj mohou být označeny data a stejně tak stavy aplikace (pokud je lze popsat konkrétními daty). Všechny zdroje jsou identifikovány vlastním URI (Uniform Resource Identifier) a REST definuje čtyři základní metody pro přístup k nim. REST využívá HTTP požadavky k odesílání dat (vytváření nebo aktualizace) metodami POST a PUT, čtení dat (dotazování) metodou GET a mazání metodou DELETE.



Obrázek 4.6: REST API

Samotný požadavek je realizován pomocí HTTP a je odeslán na adresu reprezentovanou pomocí URL (Uniform Resource Locator). URL je speciální typ identifikátoru, který kromě identifikace zdroje poskytuje i kompletní adresu zdroje a popisuje i protokol pro přístup k danému zdroji. Těmito protokoly mohou být například http, ftp, atd. Dle URI odpovídajícího konkrétnímu zdroji REST API je provedeno zpracování dat. URI tvoří nadmnožinu URL a všechny adresy URL jsou tedy URI, ale ne všechny URI jsou URL viz (Obrázek 4.7). Obecně platí, že pokud adresa začíná *http://* jedná se o adresu URL.



Obrázek 4.7: URI/URL

Příklady URI pro příhlášení a odhlášení pomocí REST API.

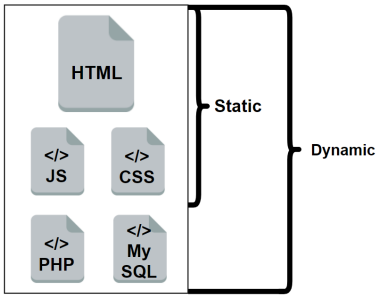
```
/api/auth/sign-in
/api/auth/sign-out
```

Využívání REST architektury má mnoho výhod. REST architektura byla vytvořena jako reakce na standardy založené na protokolu SOAP (Simple Object Access Protocol). REST klade důraz

na jednoduchou komunikaci typu point-to-point prostřednictvím HTTP. REST povoluje mnoho datových formátů jako například XML, JSON, zatímco SOAP povoluje pouze XML. Hlavním z důvodů popularity REST je skutečnost, že se jedná o rozšíření základních webových technologií jako je HTTP. Díky použití HTTP je možné jej využívat téměř na jakémkoli online zařízení. [18] [34] [35] [37]

4.3 Statický a dynamický webový server

Pro publikování webu je potřeba statický nebo dynamický webový server viz (Obrázek 4.8). Statický je označován protože odešle soubory do prohlížeče klienta tak, jak jsou vytvořeny. Jedná se o nejzákladnější druh webových stránek, který je napsán v HTML kódu, neobsahuje žádné dynamické funkce a skriptování na straně serveru. Skriptování je prováděno pouze na straně klienta. U statického serveru jsou použity zpravidla základní typy souborů (HTML, CSS, JS, obrázky). V tomto případě je každá stránka napsána v HTML zvlášť a obsah tohoto webu je možno změnit pouze změnou zdrojového kódu. Tento typ webu tedy nabízí uživateli funkce, které jsou napsány v JS. Avšak mezi jednotlivými HTTP požadavky se tento web nemění (pokud není provedena změna ve zdrojovém souboru). Tyto servery jsou zpravidla pro jednodušší aplikace.
Dynamický webový server se skládá ze statického webového serveru a dalších SW částí. U dynamického webu se jedná o web s neustále se měnícím obsahem. Hlavním rozdílem oproti statickému serveru je obvykle propojení s databází, pomocí které je možné velmi jednoduše strukturovaným způsobem získávat informace a nebo vytvářet stránky produktů. Velmi často jsou dynamické weby tvořeny pouze jako šablony, do kterých jsou vkládána data z databáze s cílem jednotného vzhledu a snadné manipulace (například obměny produktů v e-shopu). Tento typ webu vyžaduje více než jen skriptování na straně klienta a jeho součástí je i skriptování na straně serveru. Jedná se o skriptovací jazyky ASP, PHP, atd. Typickým příkladem jsou například webové stránky elektronických obchodů nebo sociálních sítí, které mohou být v některých případech tvořeny jen několika jednoduchými HTML šablonami a velkou databází. [15]



Obrázek 4.8: Rozdíl mezi statickým a dynamickým webem.

4.4 Vestavěný webový server

Vestavěné (Embedded) systémy byly zpravidla malé, samostatné, hluboce zabudované izolované systémy, které mohly komunikovat maximálně s jinými systémy v lokální síti. Tato zařízení již dnes komunikují také prostřednictvím internetu a to jak s lidmi, tak i s ostatními zařízeními. Běžné webové aplikace většinou poskytují uživateli informace nebo zábavu. Webové aplikace na vestavěných webových serverech mohou sloužit pro vzdálené ovládání či monitorování zařízení nebo systému. Vzdáleným ovládáním může být například roztočení motoru, sepnutí elektrického obvodu, otevření nebo uzavření ventilu a monitorováním například vyčtení teploty a vlhkosti. Fyzické ovládání v podobě tlačítek, otočných knoflíků a podobně bývá nahrazeno nebo duplikováno ovládáním které může pro uživatele vypadat naprosto totožně jako to fyzické s rozdílem, že je uživateli poskytnuto virtuálně ve webovém prohlížeči. V případě úplného nahrazení fyzického ovládání ovládáním virtuálním má eliminace tlačítek, otočných knoflíků, případně displejů také dopad na výrobní náklady a design samotného zařízení. Ještě důležitější aspekt u zařízení tohoto typu je jeho fyzická dostupnost. Vestavěné systémy se stále více nasazují na vzdálená nebo špatně dostupná místa. Nejsou zde vždy tedy na prvním místě finanční náklady na výrobní komponenty těchto zařízení, ale lidé, kteří se o tato zařízení musí starat a udržovat je v provozu. Toto je důvod proč je vzdálený přístup značně rozhodující při finanční rozvaze před vývojem vestavěných zařízení. Každý takový vestavěný systém tedy musí mít integrovaný webový server nebo musí mít možnost se připojit k internetu. [27]

Využitím může být například aplikace vestavěného webového serveru v meteostanicích realizovaných na platformě ESP32, které umožňují měřit teplotu vzduchu, vlhkost vzduchu nebo atmosférický tlak. V kombinaci se solárním panelem mohou být díky nízké spotřebě energie umístěny i na špatně přístupných místech a díky webovému serveru mohou být data vyčítána vzdáleně.

Dalším příkladem použití vestavěného webového serveru jsou přístroje LXI (LAN eXtensions for Instrumentation), které obvykle disponují tradičním čelním panelem. Některá zařízení však přední panel nemají a tato zařízení jsou pak ovládána například prostřednictvím webového rozhraní a je tedy umožněno je konfigurovat vzdáleně.

Kapitola 5

Webové technolgie

Tvorba webových stránek zahrnuje širokou škálu technologií a programovacích jazyků. Webové technologie se v základu dělí na dva směry. Front-end a back-end. Do front-end technologií se řadí všechny technologie, které jakýmkoli způsobem zasahují do webové stránky na straně klienta (webového prohlížeče). Tyto technologie se dále dělí do tří základních skupin:

- Sestavovací - HTML, XML, XHTML, atd.
- Stylizační - CSS
- Interagující - JavaScript, Dart, atd.

Kombinace těchto technologií umožňuje vytvořit statický web se základní funkcionalitou (interakcí s uživatelem), která je prováděna přímo na straně klienta pomocí skriptů v jazyce JavaScript.
Back-end zahrnuje více možností, protože se sem řadí téměř jakýkoli jazyk, který se dá využít na straně serveru. Nejběžněji používané jazyky jsou PHP, Java, C/C++.
Mezi základní technologie a programovací jazyky pro tvorbu na straně front-endu použité v této diplomové práci patří HTML, CSS, JS a framework Bootstrap. [36]

5.1 HTML

HTML (Hyper Text Markup Language) je hypertextový značkovací jazyk. Tento jazyk je používán k vytváření obsahu webové stránky. Tímto obsahem mohou být texty, obrázky, tabulky, atd. Jazyk HTML se skládá z řady elementů, které udávají prohlížeči jakým způsobem má zobrazit daný obsah. Těmito elementy jsou označovány části jako nadpis, obsah, odstavec a další. Jednotlivé stránky jsou mezi sebou propojeny hypertextovými odkazy. Základní kostra je vždy stejná viz (kód 5.1) a skládá se z deklarace že se jedná o HTML dokument, základního elementu *<html>*, dále z elementu *<head>* obsahujícího metadata o stránce jako je například znaková sada, odkazy na JS a CSS soubory a další. Element *title* je název stránky zobrazující se na kartě stránky. Po elementu

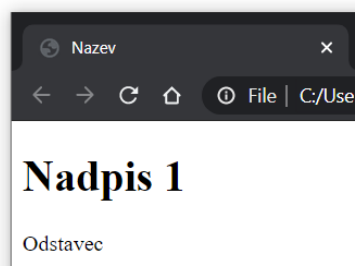
title následuje ukončení hlavičky a je definováno tělo dokumentu, které slouží jako kontejner pro všechen viditelný obsah jako jsou nadpisy, odstavce, obrázky, hypertextové odkazy, seznamy, atd. Element *<h1>* značí nadpis a element *<p>* definuje odstavec. Výsledná stránka viz (Obrázek 5.1). [30]

```
<!DOCTYPE html>
<html>
<head>
<title>Nazev</title>
</head>

<body>
<h1>Nadpis 1</h1>
<p>Odstavec</p>

</body>
</html>
```

Listing 5.1: Kostra HTML kódu



Obrázek 5.1: Webová stránka HTML kostra

Každý element je vždy definován počáteční značkou, obsahem a ukončující značkou. Úkolem webového prohlížeče (Firefox, Chrome, Safari) je číst HTML dokumenty a správně je zobrazovat. Prohlížeč nezobrazuje HTML značky, ale používá je ke správnému zobrazení dokumentu.

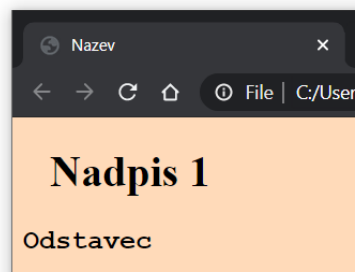
5.2 CSS

CSS (Cascading Style Sheets) je zkratkou pro stylizační šablony sloužicích ke stylizaci HTML dokumentu. CSS viz (kód 5.2) slouží k popisu jak se mají HTML elementy uživateli zobrazovat viz (Obrázek 5.2). Jeden CSS soubor může být zdrojem stylů pro více webových stránek najednou. Definování stylů, přidávání atributů jako změnu barvy, změnu velikosti písma lze realizovat i v HTML

dokumentu. HTML dokument se však takto stává postupně velmi nepřehledným, proto se vytváří zvlášť soubor CSS. [31]

```
body
{
    background-color: PeachPuff;
}
h1
{
    color: black;
    text-align: left;
    padding-left: 20px;
}
p
{
    font-family: Courier New;
    font-size: 20px;
    font-weight: bold;
}
```

Listing 5.2: CSS kód vzor



Obrázek 5.2: Webová stránka HTML kostra stylizována pomocí CSS

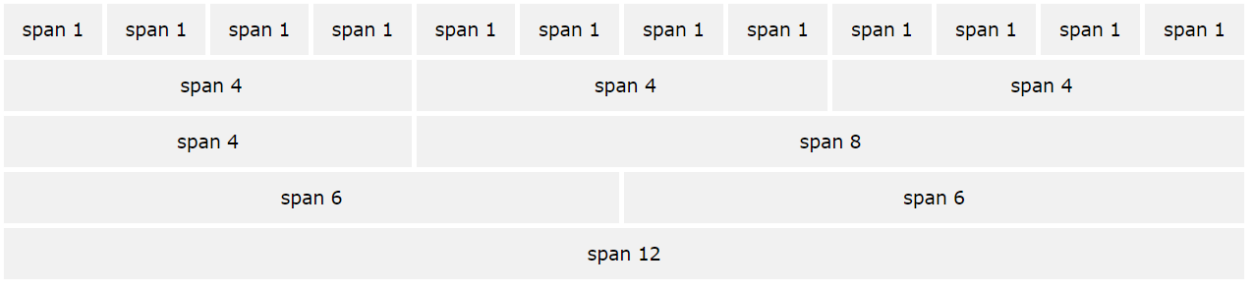
5.3 JavaScript

JS (JavaScript) je programovací jazyk určený primárně pro tvorbu webových stránek. Jedná se o jeden z nejpoužívanějších jazyků a je podporován všemi webovými prohlížeči, které díky němu nabízí příjemnější a interaktivnější uživatelské rozhraní. Tento programovací jazyk byl určen k programování klientských částí aplikací. To se však změnilo při příchodu open source prostředí *Node.js* a dalších technologií, které umožňují spuštění JavaScriptu na straně serveru. Programy napsané v

JS jsou nazývány skripty a je možné je psát přímo do HTML kódu nebo do samostatného souboru, který je následně s HTML souborem propojen. Při vykonávání JS skriptů na straně klienta není server nijak zatěžován. Díky tomu, že je kód vykonáván v prohlížeči umožňuje interakci s uživatelem, aniž by došlo k fyzickému znovunačtení stránky. Typickým příkladem může být upozornění na nevyplněnou položku ve formuláři, nebo návrh nejčastěji vyhledáváných slov již při napsání prvního písmene do vyhledávače. [32]

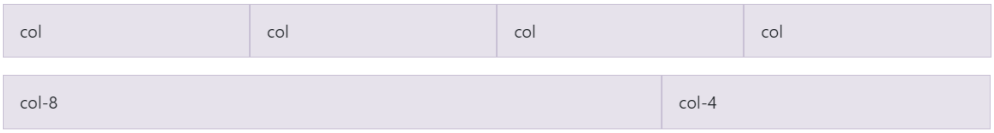
5.4 Front-end framework Bootstrap

Bootstrap je otevřená sada nástrojů kaskádových stylů pro tvorbu webu a webových aplikací. Bootstrap zahrnuje návrhářské šablony a definice stylů založené na HTML a CSS, které slouží k úpravě typografie, umožňuje výběr písma, barvy, velikosti, dále poskytuje prvky uživatelského rozhraní jako jsou dialogová okna, formuláře, tlačítka a další komponenty.
Bootstrap je využíván zejména pro tvorbu responzivních webů. Nejvýznamnější funkcí jsou možnosti rozložení stránky. Základní součást celého rozvržení stránky je tzv. kontejner do kterého jsou umístěny všechny další prvky. U tohoto kontejneru je možno si zvolit zdali bude mít pevnou šířku, kde je možné si zvolit jednu z předdefinovaných velikostí obrazovky (menší než 576 pixelů, 576–768 pixelů, 768–992 pixelů, 992–1200 pixelů, nebo větší než 1 200 pixelů), případně zvolit plovoucí šířku, kde se šířka webu bude měnit a vždy bude vyplněna celá obrazovka.
Pro možnosti rozvržení webu pomocí Bootstrapu je důležitý Bootstrap Grid System viz (Obrázek 5.3).



Obrázek 5.3: Bootstrap Grid System [29]

Tento systém umožňuje web rozdělit až na dvanáct sloupců. Pokud není požadováno 12 sloupců, je možné sloupce seskupit a vytvořit tak širší sloupce. S těmito sloupci je možné pracovat v rámci změny velikosti obrazovky. V případě otevření webu na velkém monitoru může uživateli vyhovovat například 6 sloupců. Při otevření webu na mobilním telefonu by rozdělení na 6 sloupců bylo příliš a proto je možno pro situaci s menším displejem zvolit například pouze 4 sloupce viz (Obrázky 5.4)



Obrázek 5.4: Bootstrap Grid System vzorová varianta [33]

Boostrap Grid System obsahuje tyto třídy.

- None - obrazovka šířky <576px
- sm - obrazovka šířky ≥576px
- md - obrazovka šířky ≥768px
- lg - obrazovka šířky ≥992px
- xl - obrazovka šířky ≥1200px

Kapitola 6

Výběr HW platformy pro realizaci sběrnicové brány

Na trhu existuje řada dostupných univerzálních HW platforem pro vývoj aplikací určených pro automotive, které však ve většině případů nedisponují rozhraním WiFi. Z tohoto důvodu byly při rozhodování o výběru vhodné platformy zvažovány dvě varianty.
Jednou z variant je realizace vestavěného webového serveru a konfigurační části sběrnicové brány na hardwaru, který bude nejlépe vyhovovat všem požadavkům pro konfigurační část a stejně tak realizovat řídicí část sběrnicové brány (která bude pracovat se sběrnicemi a zpracovávat data) na hardwaru, který bude nejvhodnější pro tuto část. A tyto dvě nezávislé platformy následně propojit pomocí vhodné komunikace s definovaným API.
Druhou možnou variantou bylo nalezení přijatelného kompromisu pro realizaci konfigurační i řídicí části na jednom zařízení za cenu doplnění chybějících částí externími moduly (WiFi modul, CAN kontrolér, CAN transceiver, modul čtečky SD karet, atd.). V tomto případě by propojení mezi konfigurační a řídicí částí bylo realizováno pomocí struktury ve sdílené paměti.

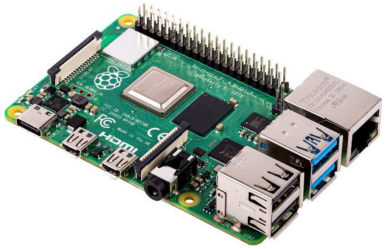
Základními požadavky při výběru HW platformy byly:

- Integrovaná WiFi (případně jednoduše doplnitelná)
- Možnost realizace CAN komunikace (případně dostupné externí moduly)
- Dostatečný výpočetní výkon
- Dostatečná SW podpora
- Příznivá cena

Vybíráno bylo mezi platformami Raspberry PI, Atmel/Microchip SAMC21, Atmel/Microchip SAME70, ST SPC5, NXP S32K, NXP Power Architecture a ESP32 od firmy Espressif Systems.

6.1 RaspberryPI

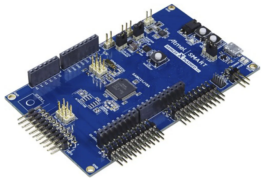
Raspberry PI 4 obsahující procesor ARM Cortex-A72 (Obrázek 6.1).
Výhody: Velmi rozšířené zařízení se značnou SW podporou, obsahuje integrovanou WiFi.
Nevýhody: Nutnost rozšíření o CAN periferie.



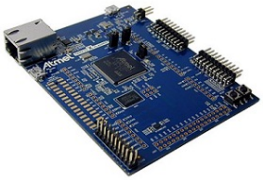
Obrázek 6.1: Raspberry PI 4 [19]

6.2 Atmel/Microchip

Z výrobků firmy Atmel/Microchip byla vhodná dvě zařízení. ATSAMC21-XPRO (Obrázek 6.2) a SAM E70 Xplained Evaluation Kit (Obrázek 6.3).



Obrázek 6.2: Atmel/ Microchip SAMC21 [20]



Obrázek 6.3: Atmel/ Microchip SAME70 [20]

6.2.1 SAMC21

ATSAMC21-XPRO je deska obsaující procesor ARM Cortex-M0+.
Výhody: 2x CAN kontrolér.
Nevýhody: Nízký výpočetní výkon procesoru, malá vnitřní paměť, nutno řešit WiFi a webový server externě.

6.2.2 SAME70

SAM E70 Xplained Evaluation Kit je vývojová deska obsahující processor ARM Cortex-M7.
Výhody: Vysoký výpočetní výkon processoru, dostatečná vnitřní paměť, 2x CAN/CANFD kontrolér.
Nevýhody: Nutno řešit WiFi se webový server externě.

6.3 NXP

Z produktů firmy NXP byla vhodná také zařízení. Prvním zařízením je NXP S32K (Obrázek 6.4) a druhým NXP Power Architecture (Obrázek 6.5).

Obrázek 6.4: NXPS32K[21]

Obrázek 6.5: MPC574XG-MB[21]

6.3.1 S32K

S32K od NXP je řada která cílí na vývoj univerzálních automobilových aplikací. Tato řada nabízí mikroprocesory ARM Cortex M0+ nebo ARM Cortex M4F. Všechny vývojové desky z této řady obsahují rozhraní pro základní automobilové sběrnice CAN a LIN.
Výhody: 3x CAN kontrolér + transceiver, LIN.
Nevýhody: Nutno řešit WiFi a webový server externě.

6.3.2 NXP Power Architecture

Nejrozšířenějším zástupcem Power Architecture od NXP je MPC574XG-MB. Jedná se o velmi spolehlivý a výkoný mikrokontrolér pro vývoj v oblasti automotive (řídicí jednotky, sběrnicové brány) a průmyslu.
Výhody: Podporuje přímé programování bez operačního systému, podporuje operační systémy jako AUTOSAR nebo Linux, 2x CAN kontrolér + transceiver, FlexRay, LIN,
Nevýhody: Nutno řešit WiFi a webový server externě. Pro cíl této diplomové práce je toto zařízení až příliš výkoné, drahé, komplikované a jeho potenciál by zůstal z naprosté části nevyužit.

6.4 ST SPC5

Vysoce výkoná a spolehlivá platforma určená pro vývoj v automotive.
Výhody: Vysoký výkon procesoru, 8x CAN/CANFD kontrolér + transceiver, FlexRay, LIN.
Nevýhody: Nutno řešit WiFi a webový server externě. Pro cíl této diplomové práce je toto zařízení až příliš výkoné, drahé, komplikované a jeho potenciál by zůstal z naprosté části nevyužit stejně jako u NXP Power Architecture.

Obrázek 6.6: ST SPC5[22]

6.5 ESP32

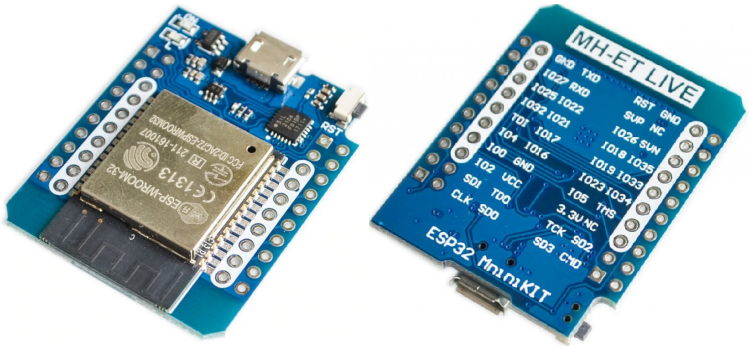
MH-ET LIVE ESP32 MiniKIT, Wemos D1 mini shield viz (Obrázek 6.7 a 6.8). Jedná se o mikrokontrolér společnosti Espressif Systems, který je nástupcem velmi populárního modelu ESP8266. Tento mikrokontrolér nedisponuje žádným operačním systémem a jeho funkcionalitu lze programovat v jazycích C, C++ nebo Wiring. ESP32 je osazen 32 bitovým dvoujádrovým mikroprocesorem LX6 Xtensa. Jedná se o mikrokontrolér, který je primárně směřován do oblasti IoT avšak díky svým parametrům, integrovanému CAN kontroléru a relativně vysokému výkonu je vhodný i pro aplikaci v mnoha jiných odvětvích.
Výhody: Vysoký výkon procesoru, integrovaná WiFi, 1x integrovaný CAN kontrolér na čipu, dostupnost, nízká cena, velká SW podpora ze strany vývojářů této platformy i ze strany komunity.
Nevýhody: Nutnost rozšíření o další CAN periferie.

6.5.1 Základní parametry

- Dvě procesorová jádra s taktem až 240 MHz
- Integrovaná paměť SRAM 520 KB
- Integrovaná WiFi se standardem IEEE 802.11 b/g/n GT40
- Integrovaný Bluetooth
- 4 MB paměti Flash
- Rozsah napájecího napětí 2.3 - 3.6 V

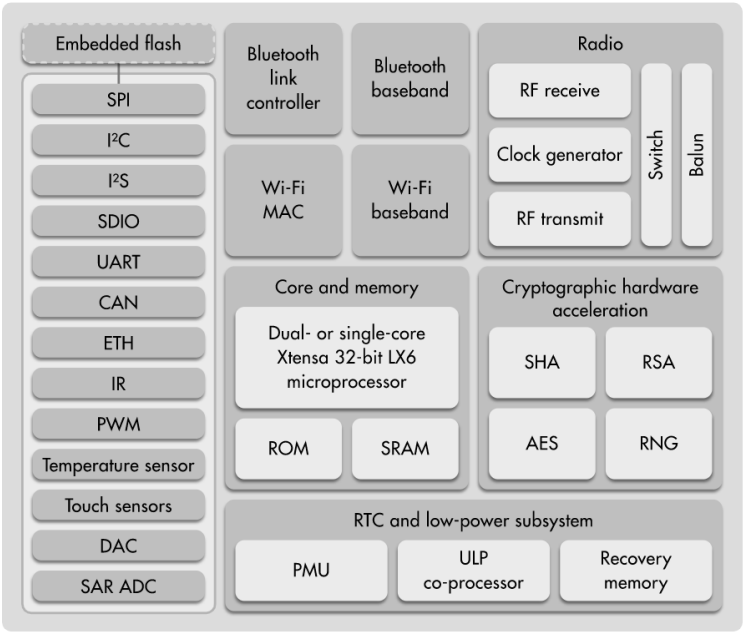
- 34 GPIO zahrnující periferie:
  - UART
  - I2C
  - ADC
  - DAC
  - I2S

Pro tuto práci byl zvolen modul ESP32 primárně z důvodu dostupnosti, integrované Wi-Fi a velké SW podpory jak ze strany výrobce, tak ze strany komunity. Komunita kolem tohoto zařízení je díky velké flexibilitě využití tohoto zařízení opravu široká. Díky tomu je dostupné velké množství knihoven a příkladů pro různé oblasti použití. Platforma ESP32 umožňuje profesionální vývoj v jazyce C a C++ s použitím IoT Development Frameworku. Pro rychlé prototypování je možné také využít prostředí Arduino IDE, které je rovněž podporováno přímo výrobcem platformy ESP32. Tato platforma nabízí velmi kvalitně zpracované a rozsáhlé knihovny pro práci s WiFi a webovým serverem. Toto zařízení nedisponuje žádnou certifikací v oblasti automotive. Cílem této práce je však zařízení pro vývoj a nejedná se o sběrnicovou bránu, která by měla být cíleně použita v automobilu v provozu. Není zde tedy certifikace vyžadována.



Obrázek 6.7: ESP32 MiniKit

Toto zařízení bylo vybráno pro realizaci konfigurační i řídicí části. Pro rozsah této práce je plně dostačující a chybějící části jako CAN kontroléry, CAN transceivery a modul čtečky paměťových karet budou doplněny periferiemi.



Obrázek 6.8: Funkční blokové schéma ESP32 [24]

Kapitola 7

Výběr vývojového prostředí pro platformu ESP32

Vývoj SW pro platformu ESP32 je možno realizovat v operačních stystémech Windows, Linux, macOS v kombinaci s několika programovacími prostředími. Eclipse IDE v kombinaci s frameworkem ESP-IDF nebo Arudino IDE ve kterých je možno programovat v programovacích jazycích C, C++ nebo Wiring. Zásluhou velké podpory uživatelů je možno využít pro programování například i IDE PlatformIO.

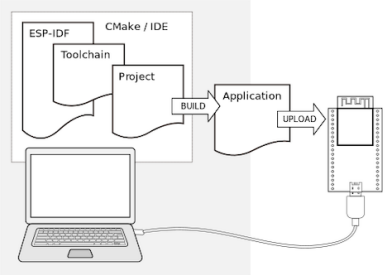
7.1 Arduino IDE

Arduino IDE je velmi populární a rozšířené programovací prostředí dostupné pro operační systémy Windows, Linux i macOS a bylo původně vyvinuté pro programování mikrokontrolérů ATmega firmy Atmel. Toto prostředí je velmi oblíbené pro svou jednoduchost. Jedná se o textový editor s kompilátorem pro velké množství mikroprocesorů. V prostředí Arduino IDE je možno psát kód v programovacím jazyce Wiring, který vychází z jazyka C++. Pro programování Arduino desek stačí pouze v nastavení zvolit použitou desku a sériový port ke kterému je deska připojena. V Arduino IDE je možno však programovat i jiné mikroprocesory. Mezi tyto mikroprocesory patří i mikroprocesor použitý u ESP32. Pro jeho programování je však nutné stáhnout potřebné soubory (vše je možné realizovat pár kroky ve správci desek Arduino IDE). Díky své popularitě a velké rozšířenosti mezi uživateli má mnoho volně dostupných knihoven napsaných od uživatelů uživatelům. [25]

7.2 ESP-IDF

ESP-IDF je framework vyvíjený výrobcem modulu ESP32, firmou Espressif Systems. Jedná se o otevřený framework obsahující možnosti základní konfigurace pro kompilaci zdrojového kódu, základní konfigurace obvodu ESP32, firmware downloader a velké množství knihoven specifických funkcí. Pro

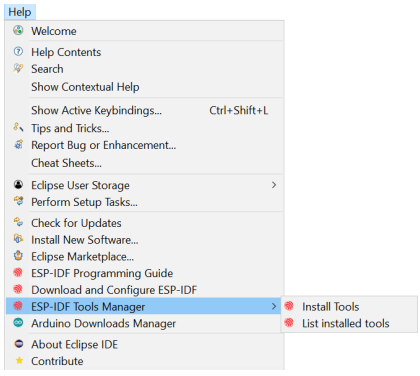
podporu použití v různých vývojových prostředích využivá ESP-IDF od verze 4.0 CMake (Obrázek 7.1), který byl vyvinut jako odezva na poptávku po multiplatformím prostředí pro kompilaci softwaru. Na základě CMake je tedy možno ESP-IDF využít v kombinaci s vývojovými prostředími a editory jako například Visual Studio Code, Eclipse, Code::Blocks a další. Do verze 4.0 byl v tomto frameworku využíván sestavovací systém GNU Make. [23]



Obrázek 7.1: Řetězec pro vývoj aplikace pro ESP32[23]

7.3 Eclipse IDE + ESP-IDF

Aduino IDE je pro základní programování na platformě ESP32 rozhodně jednodušší a přívětivější cesta a to jak ve smyslu instalace a nastavení prostředí pro práci, tak i při samotném programování. Arduino IDE ve své jednoduchosti skrývá spoustu nedostatků oproti ESP-IDF jako například umělé omezení na formu setup and loop, menší možnost konfigurovatelnosti kódu, omezené možnosti pro práci s přerušeními atd. Hlavní výhodou frameworku ESP-IDF je tedy možnost vytvářet aplikace na nejnižší úrovni a proto je vhodný zejména pro rozsáhlejší a složitější projekty. Jeho nevýhodou jsou vyšší nároky na znalosti a zkušenosti programátora a složitější nastavení programovacího prostředí. Pro zachování možnosti využívat veškeré funkce platformy ESP32 které poskytuje byla pro tuto diplomovou práci zvolena kombinace ESP-IDF a Eclipse IDE. ESP-IDF lze přidat jako plugin do Eclipse IDE (Obrázek 7.2). [23]

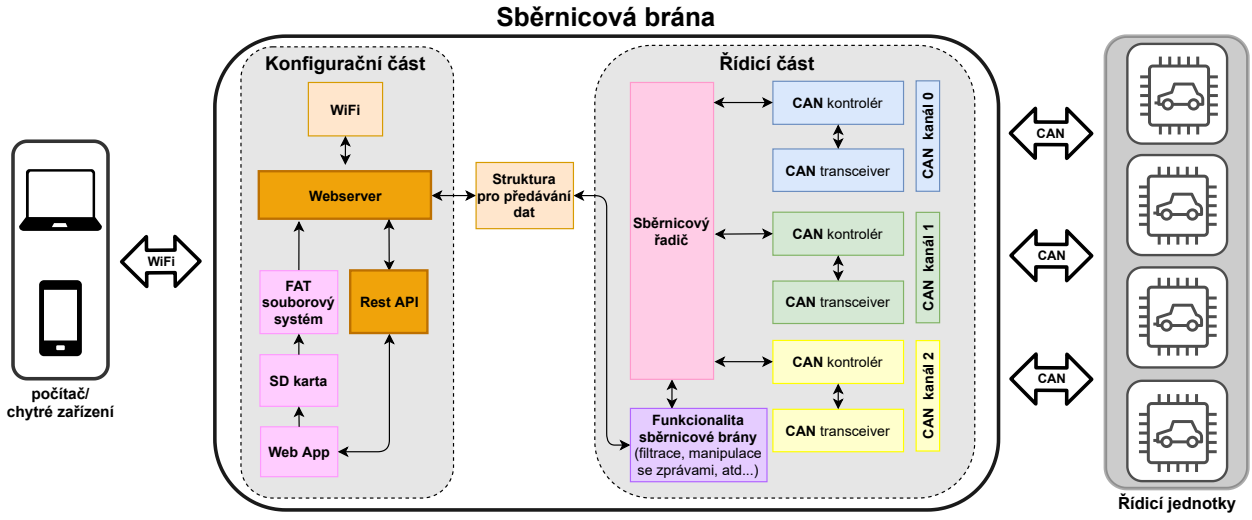


Obrázek 7.2: ESP-IDF plugin v prostředí Eclipse

Kapitola 8

Systémová architektura sběrnicové brány

Univerzální sběrnicová brána se skládá ze dvou základních celků viz (Obrázek 8.1). Prvním z těchto celků je řídicí část, která není náplní této práce a byla realizována v rámci firmy. Tato část zahrnuje samotnou funkcionalitu sběrnicové brány (filtrace zpráv, manipulace se zprávami a umožňuje komunikaci po sběrnicích). Obsahuje sběrnicový řadič a 3 CAN kanály, kde každý kanál je tvořen jedním CAN kontrolérem a jedním CAN transceiverem.
Druhým celkem je část konfigurační, která je náplní této práce. Konfigurační rozhraní sběrnicové brány je realizováno prostřednictvím webové aplikace běžící na vestavěném webovém serveru. Soubory webu jsou uloženy na SD kartě. Webová aplikace a webový server spolu komunikují pomocí JSON REST API a konfigurační a řídicí část sběrnice spolu komunikují prostřednictvím globální struktury v jazyce C.

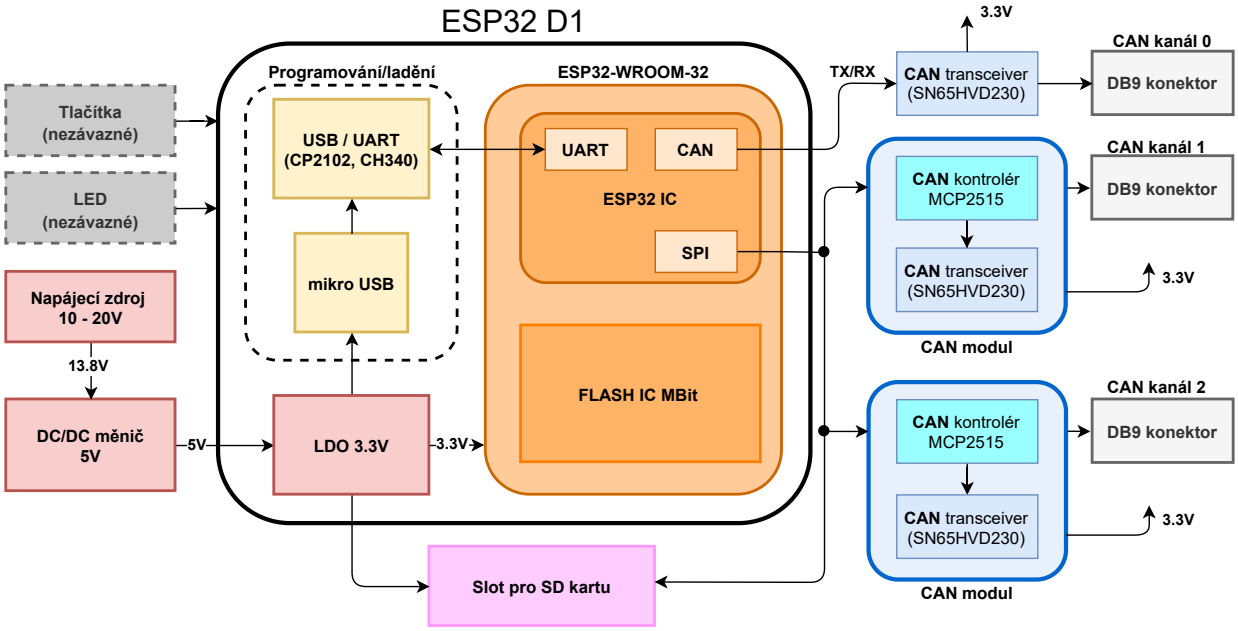


Obrázek 8.1: Systémová architektura

Kapitola 9

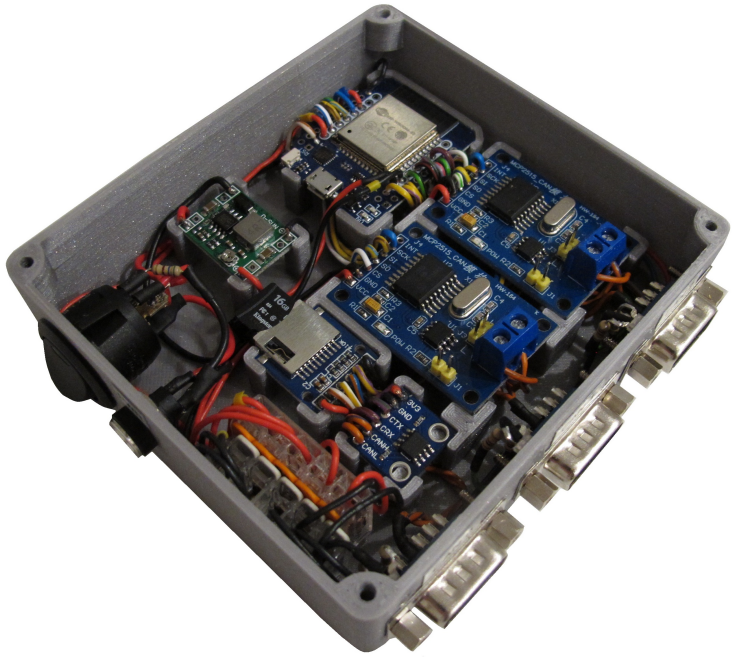
HW architektura sběrnicové brány

HW architektura sběrnicové brány viz (Obrázek 9.1) je rozdělena na několik základních částí. První částí je napájecí segment, který zahrnuje napájecí zdroj (palubní napětí v automobilu) u kterého se předpokládá velikost 10-20V, nejčastěji však hodnota přibližně 13,8 V. Pro napájení sběrnicové brány je toto napětí sníženo DC/DC měničem na napětí o hodnotě 5V a dále pomocí regulátoru napětí s nízkým úbytkem (LDO) na 3,3 V. Napětím 3,3 V jsou napájeny všechny části sběrnicové brány. Sběrnicová brána dále obsahuje konektor typu mikro USB a převodník USB/UART pro programování a ladění SW. Samotný modul ESP32-WROOM-32 zahrnuje Flash paměť o velikosti 4MB a integrovaný obvod ESP32. Pro uložení dat konfigurační části je zde použit slot pro kartu typu mikro SD. Komunikace mezi mikro SD kartou a obvodem ESP32 probíhá prostřednictvím rozhraní SPI. Sběrnicová část disponuje třemi CAN kanály. Jeden z nich zahrnuje CAN kontrolér integrovaný v obvodu ESP32 a externí CAN transceiver zakončený konektorem DB9. Pro zbylé dva CAN kanály zde nejsou integrované CAN kontroléry v obvodu ESP32 a jsou proto použity externí, které jsou propojeny s ESP32 prostřednictvím rozhraní SPI. Dále je zde pro každý z kanálů použit externí transceiver zakončený konektorem DB9.



Obrázek 9.1: HW architektura

První prototyp (návrh krabičky a osazení HW) vyvinut ve firmě viz (Obrázek 9.2).



Obrázek 9.2: První prototyp sběrnicové brány

Kapitola 10

Návrh jednotlivých funkcionalit sběrnicové brány

V konfigurační části sběrnicové brány je možno nastavit parametry kanálů a samotnou funkcionalitu sběrnicové brány sestavenou z filtrů a akcí.

10.1 Kanály sběrnicové brány

Mezi požadované funkcionality sběrnicové brány patří možnost nastavení až tří CAN kanálů současně. U každého z těchto kanálů (který bude použit) je vyžadována volba přenosové rychlosti 125 kbit/s, 250 kbit/s, 500 kbit/s nebo 1000 kbit/s. Každý z těchto kanálů je také možno pro bránu softwarově deaktivovat (i za předpokladu, že kanál je přítomen a funkční). Řídicí část sběrnicové brány poskytuje konfigurační části informaci o aktivitě na sběrnici (zdali na sběrnici probíhá nějaká komunikace). O tom který kanál slouží jako vstup a výstup je rozhodnuto u každé funkcionality individuálně.

10.2 Filtry sběrnicové brány

Sběrnicová brána může mít přednastavených až 10 filtrů označených čísly pozic 0 - 9. Ne každý filtr musí být vždy použit. Pro každý z těchto filtrů je možnost volby jeho typu PASS (propust) nebo STOP (zádrž). V závislosti na tom jsou poté zprávy v řídicí části propouštěny nebo blokovány. Dále lze pro každý filtr nastavit jestli filtruje CAN zprávy podle ID (identifikátor) nebo podle DLC (Data Length Code). Následně je nutno zvolit hodnotu nebo rozsah hodnot. Pro případ ID filtrace je možno zvolit v rozsahu hodnot 0 - 2047 a v případě DLC 0 - 8.

10.3 Akce sběrnicové brány

Stejně jako u filtrů také akcí pro sběrnicovou bránu může být přednastaveno až 10 a jsou označeny čísly pozic 0 - 9. Sběrnicová brána v aktuální verzi umožňuje pouze akci Replace ID. Pro každou akci tohoto typu je nutné zvolit hodnotu ID na kterou bude aktuální hodnota ID změněna. Hodnota může být opět změněna na hodnoty v rozsahu 0 - 2047.

10.4 Funkce sběrnicové brány

Funkcí sběrnicové brány může být taktéž až 10. Rozdíl oproti filtrům a akcím je v tom, že všechny v danou chvíli vytvořené funkce jsou aktivní! Každá z funkcí požaduje volbu jednoho vstupního a jednoho výstupního kanálu (vstupní a výstupní kanál nemohou být totožné). Dále je po uživateli vyžadováno zvolit jeden z přednastavených filtrů nebo použít volbu BP (Bypass). Tato volba znamená, že zprávy nejsou nijak filtrovány a do následné akce prochází všechny zprávy. U akce je taktéž možno vybrat volbu bypass, což zde znamená, že není na filtrovaných zprávách provedena žádná akce a zprávy které prochází filtrem jsou beze změny odeslány na výstupní kanál. Za předpokladu, že není nastaven BP ani u filtru ani u akce, jsou příchozí zprávy filtrovány, je na nich provedena požadovaná akce a následně jsou odeslány na výstupní kanál.

10.5 Obecné funkcionality sběrnicové brány

Mezi obecné funkcionality sběrnicové brány, které přímo nesouvisí s prací se zprávami na sběrnicích patří příhlášení do samotné konfigurační části sběrnicové brány, které slouží k zamezení neoprávněného přístupu. Dále samotná aktivace sběrnicové brány. Po nastavení všech provozních parametrů je zařízení aktivováno stisknutím tlačítka start. Toto tlačítko zároveň slouží pro uložení konfigurace do paměti. Po stisknutí tlačítka start jsou všechny nastavené parametry odeslány z webu na webový server ESP32 a uloženy do definované struktury kde k těmto datům má přístup řídicí část sběrnicové brány. V záložce nastavení týkající se technických parametrů HW platformy jsou uvedeny informace o obvodu ESP32 jako verze IDF, revize čipu, atd. Další z funkcionalit je odhlášení ze sběrnicové brány. Po odhlášení je uživatel přesměrován na úvodní (přihlašovací) stránku. Mezi další obecné funkcionality sběrnicové brány patří základní kontrola informací zadávaných uživatelem. Data ve kterých hrozí riziko, že uživatel může udělat chybu jsou před odesláním kontrolována na straně prohlížeče pomocí skriptů napsaných v JS. Typickým příkladem je zapisování hodnoty ID do textového pole Condition, kde je povolen rozsah hodnot pouze pro 11 bitový identifikátor, který může nabývat hodnot 0 - 2047 a je tedy nežádoucí aby uživatel do této kolonky zadával cokoli jiného (text, hodnoty mimo rozsah). V případě špatně zadaných hodnot bude uživatel upozorněn výstražným banerem a stejně tak pokud zanechá pole nevyplněné.

Kapitola 11

Návrh a implementace webové aplikace pro konfiguraci parametrů sběrnicové brány

Hlavním účelem webové aplikace vyvíjené v této DP je umožnit konfiguraci sběrnicové brány prostřednictvím libovolného mobilního zařízení nebo PC bez ohledu na použitý operační systém či velikost a rozlišení obrazovky. Tohoto požadavku lze dosáhnout použitím webových technologií jako například Front-end framework Bootstrap.

11.1 Implementace Bootstrap Front-end frameworku

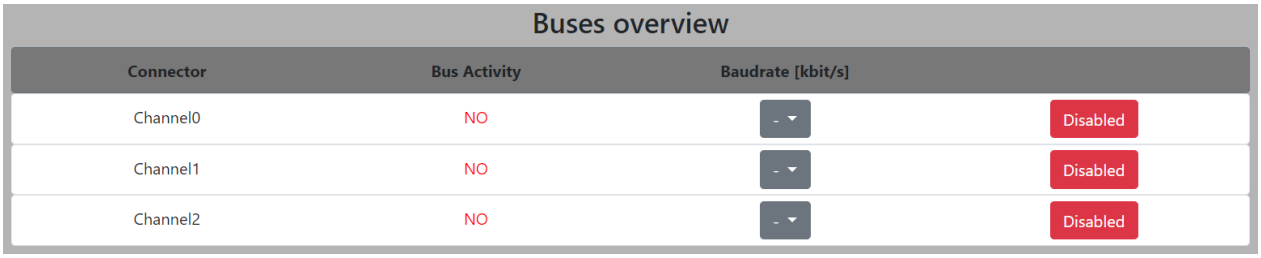
Bootstrap Grid Systém obsahuje několik tříd definující šířky obrazovky. Jedná se o třídy *none*, *sm*, *md*, *lg*, *xl* viz (5.4). V závislosti na těchto šířkách lze měnit dynamicky rozložení stránky při změně velikosti obrazovky. Část HTML kódu webu sběrnicové brány viz (kód 11.1).

```
<div class="col-3 col-sm-3 col-md-3 col-lg-3 col-xl-3 font-weight-bold">
    Connector</div>
<div class="col-2 col-sm-2 col-md-3 col-lg-3 col-xl-3 font-weight-bold">Bus
    Activity</div>
<div class="col-3 col-sm-3 col-md-3 col-lg-3 col-xl-3 font-weight-bold">Baudrate
     [kbit/s]</div>
<div class="col-4 col-sm-4 col-md-3 col-lg-3 col-xl-3 font-weight-bold"></div>
```

Listing 11.1: Bootstrap Grid System pro filtry

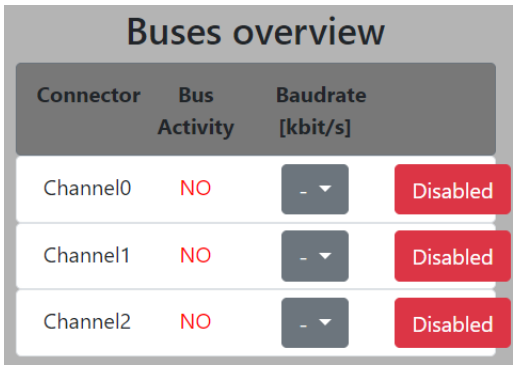
Pro třídu *col-* a třídu *col-sm* což jsou třídy pro nejmenší velikost obrazovek (mobilní telefon nebo malý tablet) je zvolena v kódu 11.1 pro první a třetí element * * (Connector a Baudrate [kbit/s]) šířka sloupce 3, pro druhý * * (Bus Activity) je zvolena šířka sloupce 2 a pro poslední * * (tlačítka pro aktivaci/deaktivaci kanálu) je zvolena šířka čtyři sloupce. Pro větší rozměry obrazovek jako jsou třídy *col-md*, *col-lg* *col-xl* jsou již pro všechny elementy * * sloupce shodně široké.

Toto rozdělení sloupců je realizováno z důvodu tlačítka umístěného v posledním <*div*> elementu viz (Obrázek 11.1).

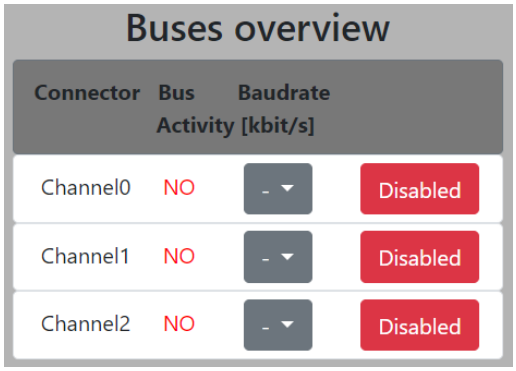


Obrázek 11.1: Buses Overview velká obrazovka col-3, 3, 3, 3

V případě zobrazení stránky na obrazovce s vyšším rozlišením (například 1536 x 864 px) viz (Obrázek 11.1) vypadá rozložení 4x3 sloupce velmi přehledně a esteticky. Kdyby toto rozložení zůstalo fixní bez ohledu na velikost obrazovky, tak by na malé obrazovce nebylo dost místa pro tlačítka, které se do rastru 4x3 sloupce při šířce obrazovky 420 px nevejdou viz (Obrázek 11.2). Při změně rozložení na velikosti 3, 2, 3, 4 vypadá vše velmi přehledně a esteticky i na obrazovce se šířkou 420 px viz (Obrázek 11.3).

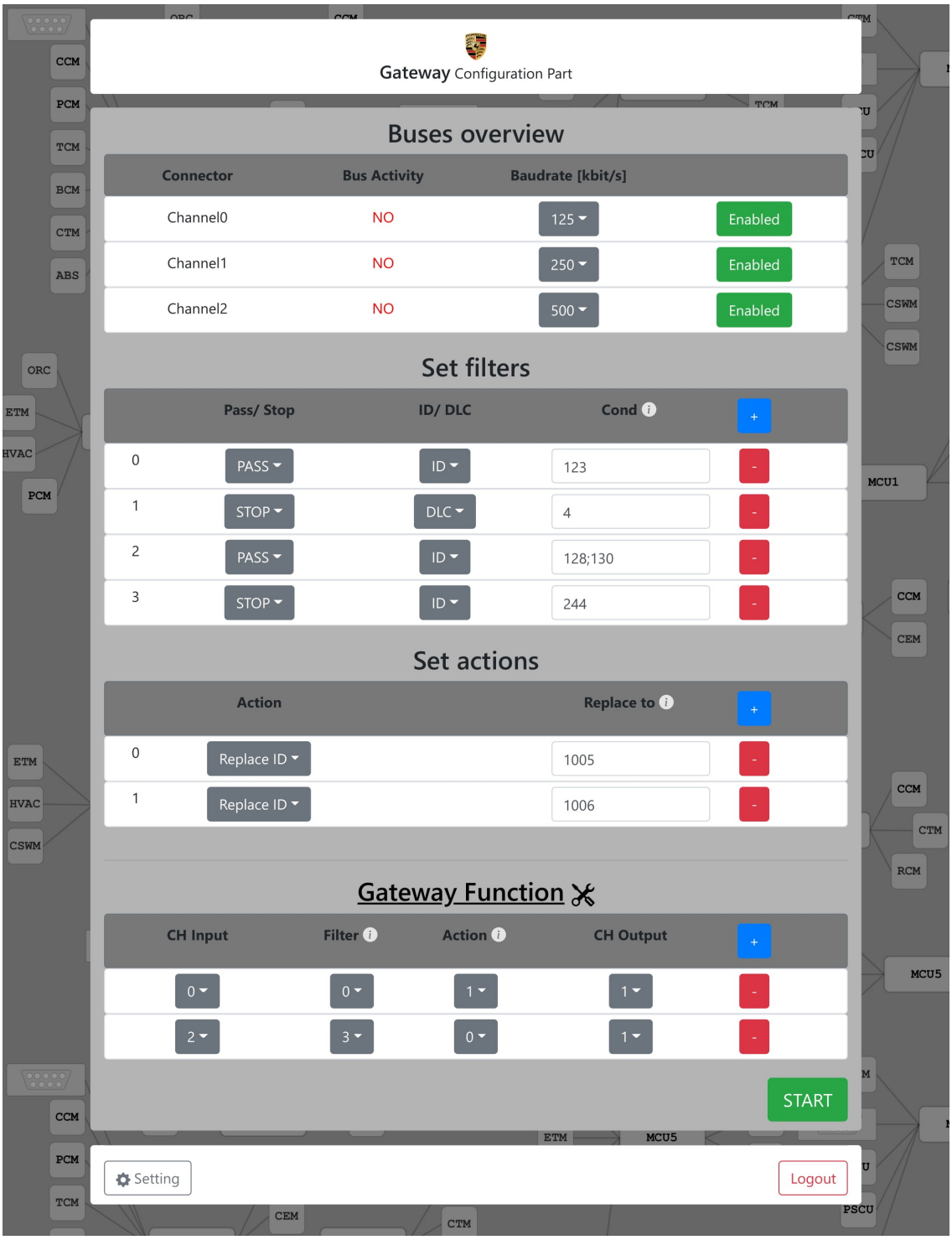


Obrázek 11.2: Šířka obrazovky 420px rozložení col-3, 3, 3, 3



Obrázek 11.3: Šířka obrazovky 420px rozložení col-3, 2, 3, 4

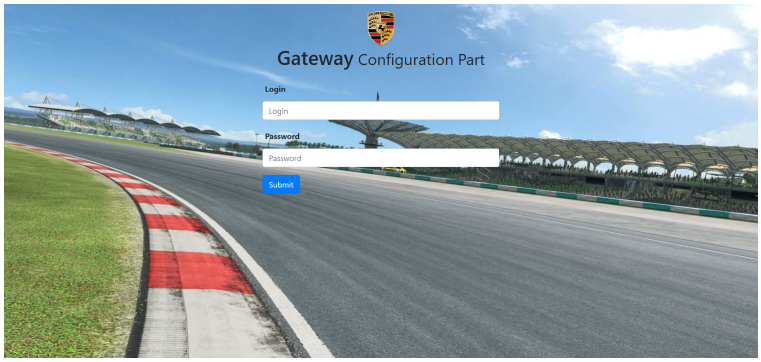
Tímto způsobem je realizován celý web s cílem umožnit jej pohodlně používat na obrazovce s téměř jakýmkoli rozlišením viz (Obrázky 11.5, 11.6, 11.7, 11.9, 11.10, 11.11). Vzhled celé hlavní stránky sběrnicové brány viz (Obrázek 11.4).



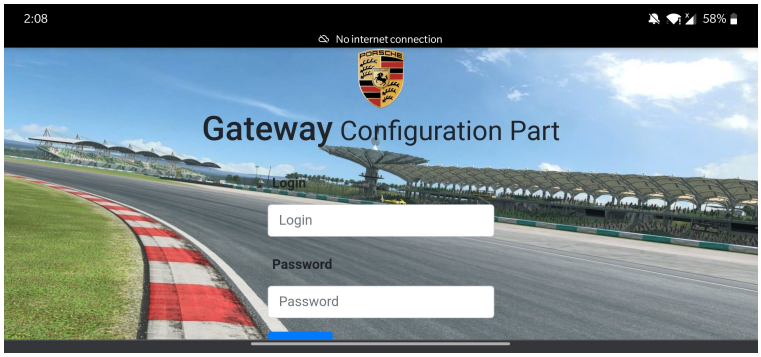
Obrázek 11.4: Celá hlavní stránka sběrnicové brány

11.2 Přihlašovací obrazovka sběrnicové brány

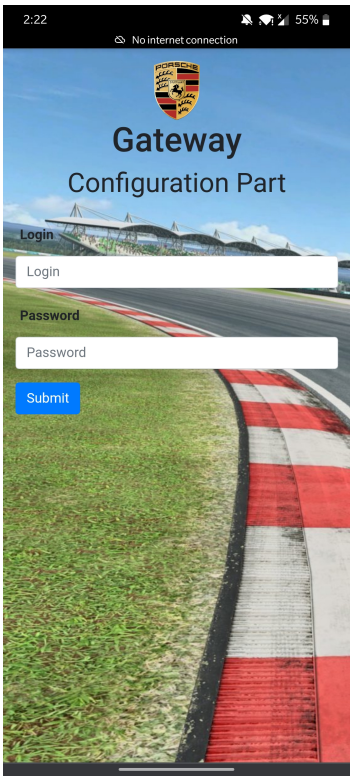
Přihlašovací stránku viz (Obrázek 11.5) bylo cílem realizovat co nejjednodušeji a obsahuje tedy kromě pozadí a loga Porsche pouze přihlašovací formulář pro vyplnění přihlašovacího jména a hesla. Po stisknutí tlačítka Submit jsou data vyčtena z formuláře a odeslána pomocí HTTP požadavku ve formátu JSON na webový server. Data jsou webovým serverem přijata, zkontrolována a odpověď (zdali je uživatelské jméno a heslo správné) je odeslána zpět na web, který uživatele přesměruje na hlavní stránku sběrnicové brány viz (Obrázek 11.9). V případě, že zadané údaje nejsou správné, uživatel je informován varovným banerem viz (Obrázek 11.8).



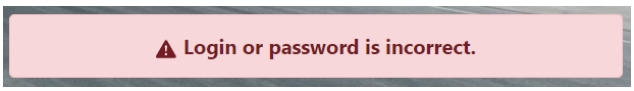
Obrázek 11.5: Přihlašovací stránka sběrnicové brány obrazovka 1536 x 864 px



Obrázek 11.6: Přihlašovací stránka sběrnicové brány obrazovka 915 x 412 px



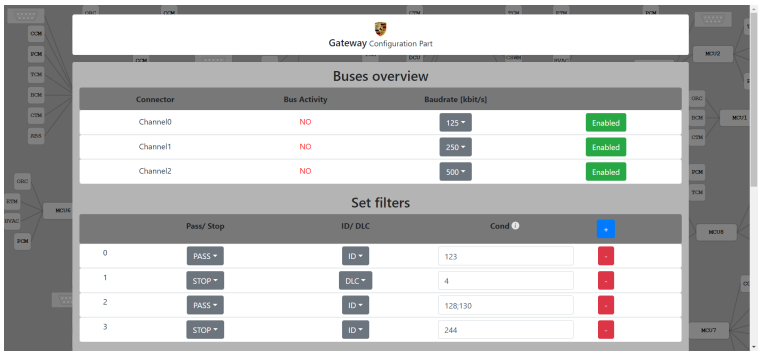
Obrázek 11.7: Přihlašovací stránka sběrnicové brány obrazovka 412 x 915 px



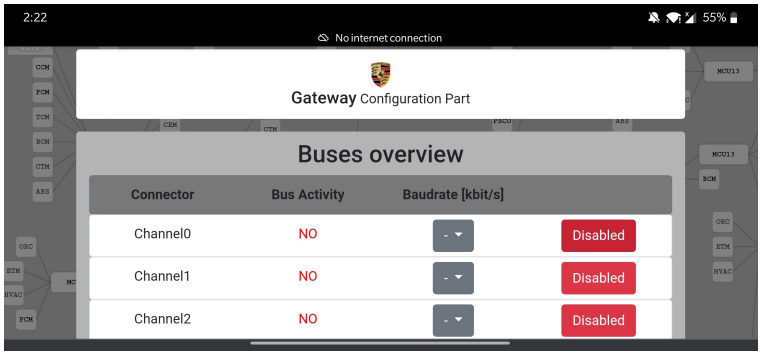
Obrázek 11.8: Varovný baner zobrazený při zadání špatných přihlašovacích údajů.

11.3 Hlavní stránka sběrnicové brány

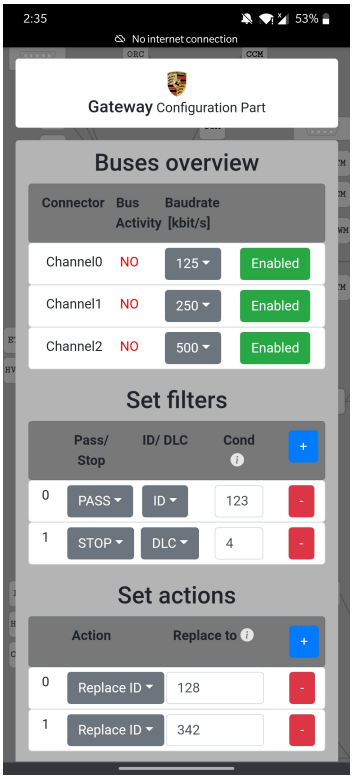
Po úspěšném přihlášení je uživatel přesměrován na hlavní stránku sběrnicové brány viz (Obrázek 11.9), která byla graficky rozdělena na tři základní části. První částí je záhlaví, které obsahuje pouze nadpis a logo Porsche. V tomto rozvržení je však vhodně předpřipraveno jako místo pro umístění lišty s tlačítky pro přepínání mezi dalšími okny.
Další menší část tvoří zápatí, které bylo realizováno jako oddělený prostor pro funkce jako je odhlášení nebo nastavení, které zde slouží k zobrazení obecných informací o zařízení a jako prostor pro případné budoucí funkce související s obecným nastavením sběrnicové brány.
Poslední a také největší část je tvořena samotným nastavením parametrů funkcí sběrnicové brány ve smyslu manipulace se zprávami na sběrnicích. Tato část byla také rozdělena systematicky na čtyři podčásti, které jsou od sebe graficky odděleny tak, aby uživatelské prostředí bylo přehledné.
Za částí sloužící k nastavení parametrů funkcí následuje pouze tlačítko START/STOP, po jehož stisku dojde k vyvolání požadavku odeslání všech nastavených dat na webový server. Po úspěšném spuštění jsou všechna tlačítka, textová pole, atd. uvedena do stavu ve kterém není možné s nimi manipulovat.



Obrázek 11.9: Hlavní stránka sběrnicové brány obrazovka 1536 x 864 px



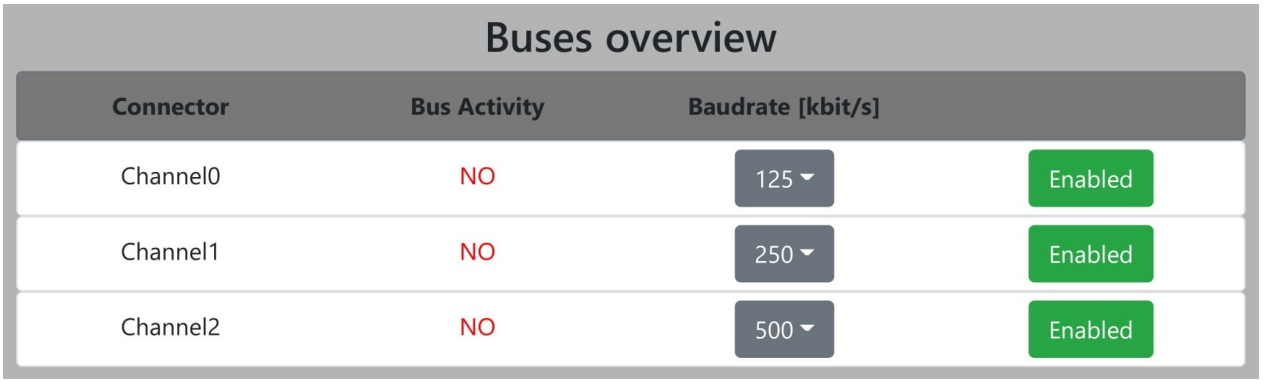
Obrázek 11.10: Hlavní stránka sběrnicové brány obrazovka 915 x 412 px



Obrázek 11.11: Hlavní stránka sběrnicové brány obrazovka 412 x 915 px

11.3.1 Přehled a nastavení sběrnic

V horní části hlavní stránky sběrnicové brány se nachází přehled sběrnic viz (Obrázek 11.12). Byl zde pevně předdefinován počet CAN kanálů na 3 (protože tato sběrnicová brána obsahuje fyzicky pouze 3 CAN konektory) a záleží pouze na uživateli zda bude využívat všechny, nebo si vystačí pouze se dvěmi. Pro každý kanál byl vytvořen indikátor aktivity na sběrnici (informace zda se po sběrnici posílají nějaké zprávy a sběrnice není nefunkční/odpojená). Tento indikátor si pomocí HTTP požadavku vyžádá informace o aktivitě na sběrnici při připojení mobilního telefonu nebo počítače ke sběrnicové bráně a následně se aktualizuje s každým obnovením stránky. Pro každý z kanálů je po uživateli požadováno aby zadal přenosovou rychlost. Podporované přenosové rychlosti jsou 125 kbit/s, 250 kbit/s, 500 kbit/s a 1000 kbit/s. Poslední parametr kanálů ve sběrnicové části je možnost aktivovat a deaktivovat jednotlivé kanály.

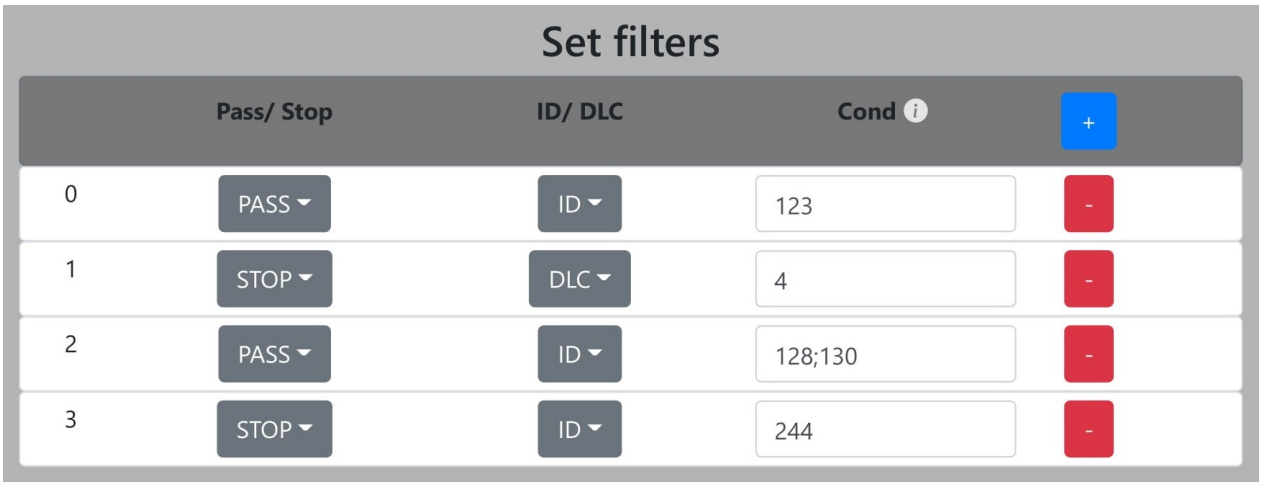


Obrázek 11.12: Přehled a nastavení sběrnic

11.3.2 Nastavení filtrů

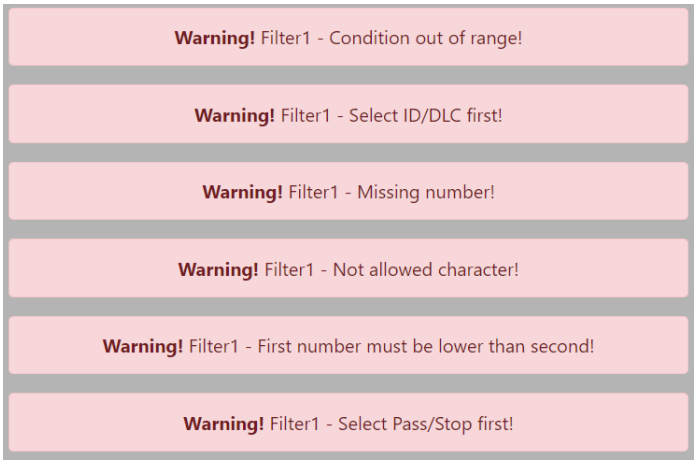
Za sběrnicovou částí následuje část pro nastavení filtrů viz (Obrázek 11.13). Je možno zde nastavit až 10 filtrů. V základním stavu je seznam prázdný, aby nebyla stránka zbytečně obsáhlá, když nejsou vytvořeny žádné filtry. Pomocí modrého tlačítka " + " v pravém horním rohu je umožněno uživateli přidat až 10 filtrů a každý si nastavit podle svých požadavků. Každý přidaný filtr musí být řádně vyplněn, jinak je uživatel upozorněn varovným banerem viz (Obrázek 11.14). Filtry jsou indexovány čísly 0 - 9. Vytvořené filtry je možno libovolně mazat červeným tlačítkem " - " u každého z filtrů. Při tomto mazání se nemění čísla kterými jsou jednotlivé filtry označeny při jejich vytvoření, aby nedocházelo k chybám způsobeným posunutím pořadí filtrů. U každého filtru je nutné nastavit parametr PASS/STOP podle kterého je určeno zda jsou dané zprávy propouštěny nebo blokovány. Dále je nutné nastavit parametr ID/DLC, který udává jestli je filtrace prováděna na základě velikosti identifikátoru nebo velikosti délky dat. U obou těchto parametrů uživatel volí pouze pomocí rozklikávacího tlačítka, které nabízí pouze tyto konkrétní dvě možnosti a je tedy eliminováno možné způsobení chyby (například při zapisování volby textově). Poslední částí, která

je po uživateli požadována je samotné číslo nebo rozsah ID nebo DLC, který má být filtrován. V případě jednoho konkrétního ID nebo DLC je po uživateli požadováno pouze jedno číslo, v případě kdy chce uživatel filtrovat určitý rozsah čísel, je požadováno aby zadal dvě čísla oddělená středníkem. Povolený rozsah pro identifikátor je 0 - 2047 a pro DLC 0 - 8. V případě nejasností je pro uživatele dostupná nápověda přímo nad daným sloupcem viz (Obrázek 11.15).

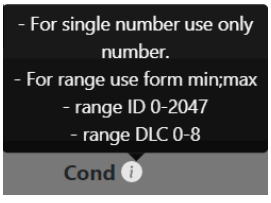


Obrázek 11.13: Nastavení filtrů

Pro zabezpečení, že se z konfigurační části sběrnicové brány do řídicí části nedostane žádná chyba jako například zapomenutá volba PASS/STOP, zapomenutá volba ID/DLC, zadaná hodnota ID nebo DLC mimo rozsah, atd. Jsou všechna tato data před odesláním na webový server zkontrolována a případně je akce odeslání přerušena a uživatel je upozorněn některým z výstražných banerů s informací ve které části se nachází chyba nebo chybějící údaje viz (Obrázek 11.14).



Obrázek 11.14: Seznam chyb odchytávaných při nastavování filtrů



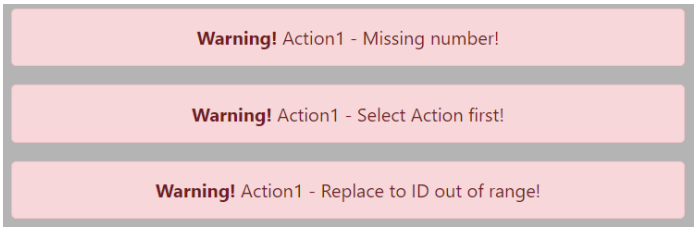
Obrázek 11.15: Nápověda pro zvolení čísla/rozsahu ID nebo DLC filtru

11.3.3 Nastavení akcí

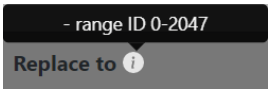
Další částí hlavní stránky sběrnicové brány je nastavení akcí viz (Obrázek 11.16). Těchto akcí může být uživatelem nastaveno opět až 10. V základním stavu je seznam prázdný. Pomocí modrého tlačítka " + " je možno přidat jednotlivé akce a pomocí červeného tlačítka " - " mazat, stejně jako v případě filtrů. Pro vytvořené akce je po uživateli vyžadováno aby byly kompletně a správně vyplněny. Jedná se zde o volbu akce, kde je prozatím k dispozici pouze možnost změny identifikátoru zprávy a o volbu hodnoty na kterou bude původní identifikátor změněn.

Obrázek 11.16: Nastavení akcí

I zde jsou všechny vstupy kontrolovány před odesláním dat na webový server pro zabezpečení, že se z konfigurační části do části řídicí nedostanou žádné chybné vstupy. V případě zjištění problému je uživatel upozorněn výstražným banerem viz (Obrázek 11.17) a odeslání je přerušeno. U akcí je kontrolováno pouze zda je zvolena akce, zda je zadáno číslo identifikátoru na který má být aktuální číslo změněno a zda je toto zadané číslo v povoleném rozsahu 0 - 2047 na což je opět uživatel upozorněn i v nápovědě k danému textovému poli viz (Obrázek 11.18). Přítomnost nedovolených znaků zde není kontrolována, protože tento *input* prvek je v HTML definován jako typ *number* a jiné znaky než čísla do něj nelze zapsat.



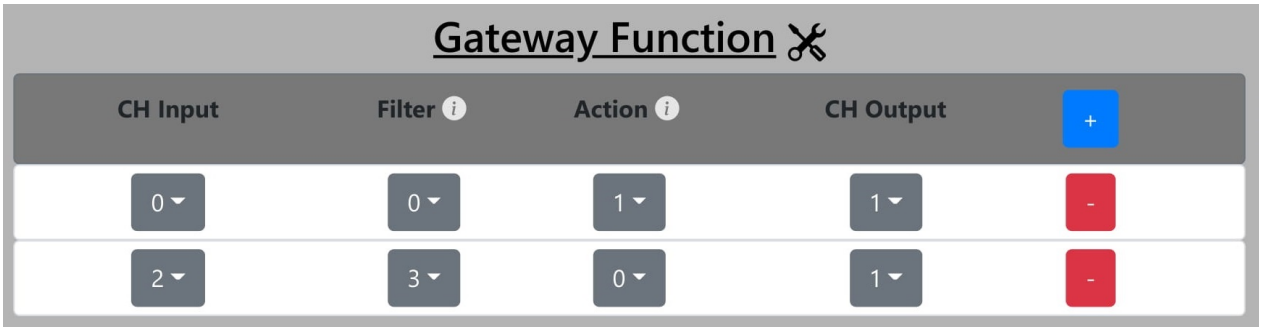
Obrázek 11.17: Seznam chyb odchytávaných při nastavování akcí



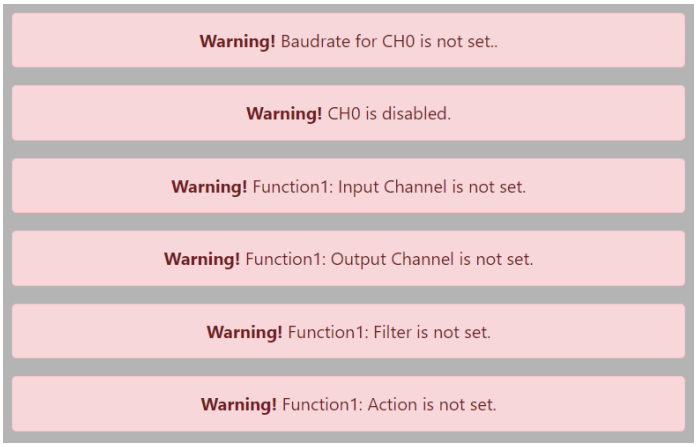
Obrázek 11.18: Nápověda pro zvolení čísla ReplaceID

11.3.4 Sestavení funkcí sběrnicové brány

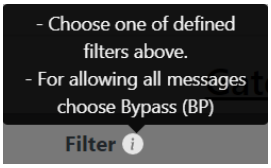
Poslední částí na hlavní stránce sběrnicové brány je samotné sestavení funkcí viz (Obrázek 11.19). Funkcí může být vytvořeno až 10 a všechny jsou automaticky aktivní. Přidání funkce je realizováno pomocí modrého tlačítka " + " a odstranění pomocí červeného tlačítka " - " u každé z nich. Každá se skládá ze čtyř parametrů.
Prvním z parametrů je vstupní kanál. Vstupní a výstupní kanál nesmí být totožný, proto je po volbě vstupního kanálu 0, 1 nebo 2 zvolený vstupní kanál automaticky odstraněn z nabídky volby výstupního kanálu, aby nemohl být zvolen. Každý musí mít nastavenou komunikační rychlost a musí být ve stavu enabled, jinak dochází k upozornění uživatele výstražným banerem.
Druhým parametrem je filtr. Zde uživatel zvolí jeden z filtrů (přednastavených v části nastavení filtrů viz kapitola 11.3.2). Uživateli jsou dány k výběru pouze přednastavené filtry nebo volba BP (Bypass). Ostatní jsou nedostupné aby bylo zamezeno nechtěnému zvolení nedefinovaného filtru. Při zvolení varianty BP se funkce chová jakoby neobsahovala žádný filtr a byla plně průchozí (propouští ze vstupu na výstup všechny zprávy). Ke sloupci pro volbu filtru je uživateli poskytnuta nápověda viz (Obrázek 11.21).
Třetím parametrem je volba akce. Zde uživatel stejně jako u filtru zvolí jednu z akcí (přednastavených v části nastavení akcí viz kapitola 11.3.3). Uživateli jsou dány k výběru pouze přednastavené akce nebo volba BP (Bypass). Ostatní akce jsou nedostupné aby bylo zamezeno nechtěnému zvolení nedefinované akce. Při výběru varianty BP se funkce chová jakoby neobsahovala žádnou akci a příchozí zprávy jsou pouze filtrovány a bez zásahu do identifikátoru odesílány na výstup. Ke sloupci pro volbu akce je uživateli poskytnuta nápověda viz (Obrázek 11.22).
Čtvrtý parametrem pro nastavení funkce sběrnicové brány je výstupní kanál. Vstupní a výstupní kanál nesmí být totožný, toto je však ošetřeno již ve vstupním kanálu, který zablokuje možnost zvolit na výstupu totožný kanál. V případě, kdy je zvolen jako první výstupní kanál a až následně vstupní kanál, je prioritizován vstupní kanál a výstupní kanál je uveden do stavu "-"(nezvolen). Zvolený výstupní kanál musí mít stejně jako kanál vstupní nastavenou komunikační rychlost a musí být ve stavu enabled.
Při nezvolení kteréhokoli z požadovaných parametrů je uživatel upozorněn výstražným banerem viz (Obrázek 11.20) a není povoleno sběrnicovou bránu spustit.



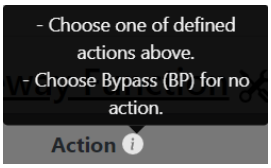
Obrázek 11.19: Sestavení funkcí sběrnicové brány



Obrázek 11.20: Seznam chyb odchytávaných při sestavování funkcí

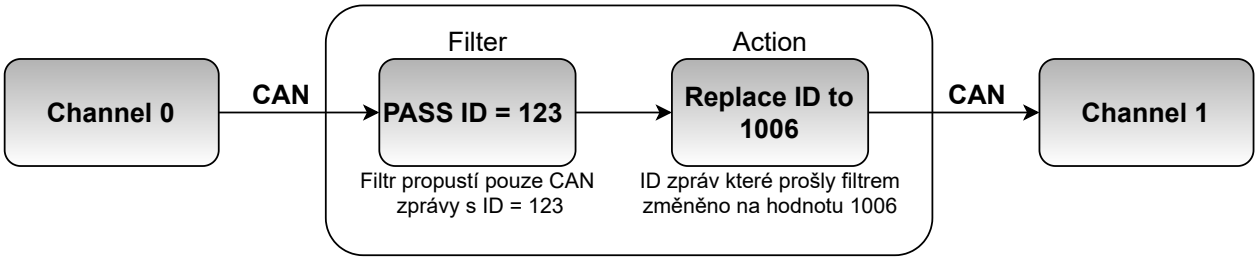


Obrázek 11.21: Nápověda pro zvolení filtru funkce



Obrázek 11.22: Nápověda pro zvolení akce funkce

Požadavkem může být například propouštět z kanálu 0 pouze CAN zprávy s identifikátorem který se bude rovnat hodnotě 123, všem těmto zprávám změnit identifikátor na hodnotu 1006 a následně tyto zprávy odeslat na kanál 1 viz (Obrázek 11.23).



Obrázek 11.23: Příklad použití sběrnicové brány

Kapitola 12

Realizace WiFi přístupového bodu a vyčítání souborů webu z SD karty

Mikrokontrolér ESP32 nedisponuje operačním systémem a pro realizaci kódu WiFi přístupového bodu a vyčítání souborů z SD karty bylo využito kombinace frameworku ESP-IDF a vývojového prostředí Eclipse IDE. ESP-IDF obsahuje základní knihovnu <esp_wifi.h> umožňující realizaci WiFi konektivity na platformě ESP32. Tato knihovna poskytuje podporu spuštění, nastavení a sledování funkcí WiFi. Dále umožňuje nastavení pro režim klienta (Station Mode), kdy se obvod ESP32 připojuje k přístupovému bodu. Dále nabízí režim přístupového bodu (Access Point), kdy je ESP32 jako přístupový bod a stanice se k němu připojují. Poslední variantou je kombinovaný režim, kdy je obvod ESP32 současně přístupový bod a současně stanice připojená k jinému zařízení.
Funkce main pro spuštění a nastavení WiFi, získání dat použitých pro realizaci webu a samotné nastavení a spuštění webového serveru implementujícího REST API je rozdělena do tří základních funkcí viz (Kód 12.1).

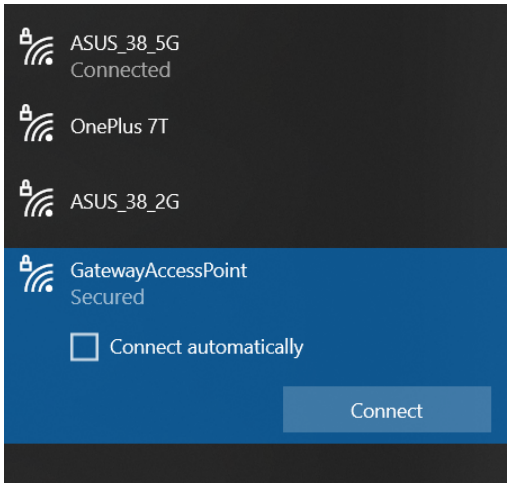
```
void app_main(void)
{
    ESP_ERROR_CHECK(wifi_init_softap());
    init_sd_card();
    start_rest_server("/PorscheGateway");
}
```

Listing 12.1: Main funkce v kódu C

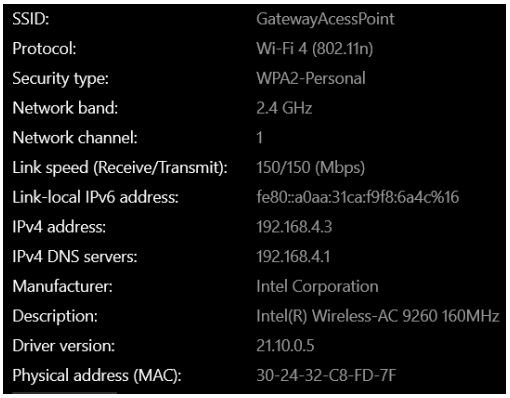
12.1 WiFi přístupový bod

Pro tuto práci byla WiFi na platformě ESP32 použita v režimu přístupového bodu (Access Point). Pro WiFi byl definován: název, heslo a maximální počet zařízení které mohou být v jeden okamžik

připojeny. Samotná inicializace AP obsahuje inicializaci základního TCP/IP stacku (TCP/IP stack je sada komunikačních protokolů používaných internetem). Dále inicializaci NVS (Non-volatile storage) , spuštění výchozí událostní smyčky (speciální typ smyčky používaný pro systémové události, jako jsou například události WiFi), alokace zdrojů pro WiFi (struktura ovládání WiFi, vyrovnávací paměť RX/TX), registrace obslužných rutin do smyčky systémových událostí, přiřazení maker do WiFi konfigurace (jméno, heslo, maximální počet zařízení) a přiřazení autorizačního módu WIFI_AUTH_WPA_WPA2_PSK. WPA2-PSK označované také jako WPA - Personal na rozdíl od WPA - Enterprise nevyžaduje autentizační server. Pro autentizaci je zde použito PSK (Pre-shared key) neboli předsdílený klíč což je heslová fráze, která je uložena na straně přístupového bodu a stejně tak na straně klienta ještě před zahájením komunikace. Nakonec je zvolen mód ve kterém bude WiFi pracovat (pro tuto práci zvolen WIFI_MODE_AP). Následně se WiFi nastaví dle navolené konfigurace a spustí se. IP adresa tohoto zařízení je nastavena jako 192.168.4.1.
WiFi je poté viditelná pro všechny okolní zařízení a je možné se k ní připojit viz (Obrázek 12.1). Parametry vytvořené sítě viz (Obrázek 12.2).



Obrázek 12.1: Vytvořená síť

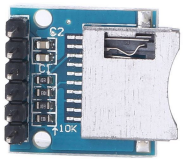


Obrázek 12.2: Parametry sítě

12.2 SD karta

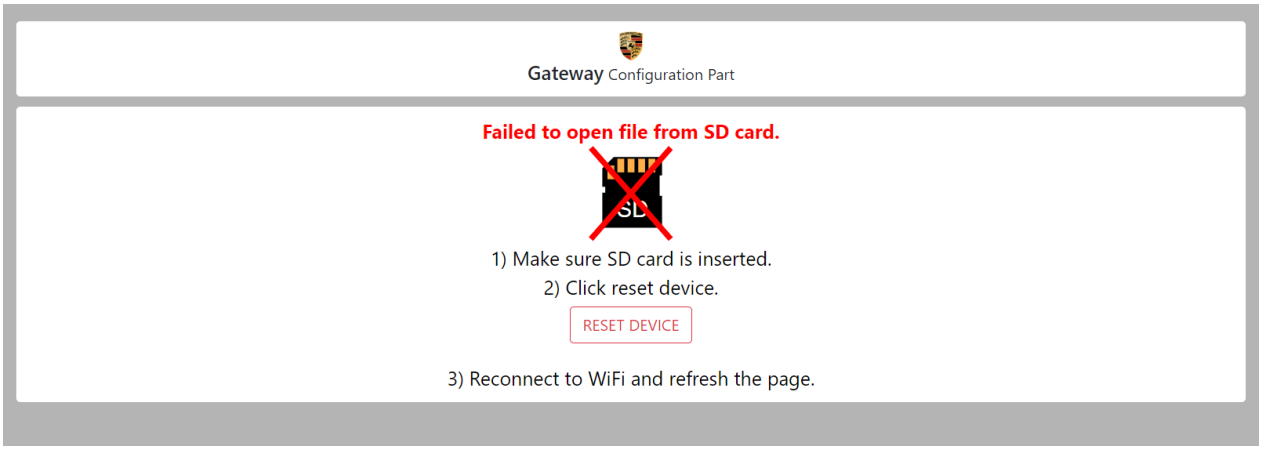
Během realizaci první verze webu nastal problém s velikostí vnitřní paměti obvodu ESP32. Flash paměť ESP32 má velikost 4 MB. Pro samotný program uživatele a data zbývají přibližně 2 MB. Tato paměť je dostatečná pro uložení základních HTML, CSS, JS souborů i některých obrázků (vhodné využívat vektorové obrázky z důvodu šetření paměti). Při použití větších nebo rastrových obrázků paměť již není dostatečná a při případném rozšiřování webu by nemusela stačit ani pro samotné soubory tvořící web. Proto byla zvolena již ze začátku cesta ukládání souborů potřebných

pro realizaci webu na mikro SD kartu, aby nebylo nutné k tomuto řešení přecházet dodatečně. Použit byl modul čtečky mikro SD karet (Obrázek 12.3) s komunikačním rozhraním SPI a napájením 3.3V.



Obrázek 12.3: Modul čtečky mikro SD karet [28]

Komunikace mezi mikro SD kartou a platformou ESP32 je realizována prostřednictvím rozhraní SPI. Jsou zde nastaveny parametry jako rychlost komunikace, piny ke kterým je modul SD karty připojen atd. Dále je provedena inicializace SPI ovladače a SD karty dle nastavených parametrů a získání souborového systému FAT (File Allocation Table) pro kartu registrovanou ve VFS (Virtual File System). Pokud by nastala chyba při inicializaci SD karty a nebylo by z ní možné vyčítat soubory, uživatel nemá možnost bez programovacího prostředí tento problém v danou chvíli zjistit. Při pokusu o načtení webové stránky (v případě neošetření možné chyby inicializace SD karty) není uživateli vráceno nic a uživatel vidí pouze prázdnou bílou webovou stránku.
Z tohoto důvodu byl do vnitřní paměti obvodu ESP32 implementován velmi malý web pro jednoduché a pohodlné řešení tohoto problému. Tento web je načten v okamžiku, kdy není vložena SD karta, je vložena špatně, nebo nastane jiný problém při vyčítání dat a uživatele na tento problém upozorní a pomůže mu jej v několika jednoduchých krocích vyřešit viz (Obrázek 12.4).



Obrázek 12.4: Interní web spuštěný v případě chyby při načtení SD karty

Kapitola 13

Návrh a implementace webového serveru a REST API pro komunikaci mezi webovou aplikací a webovým serverem

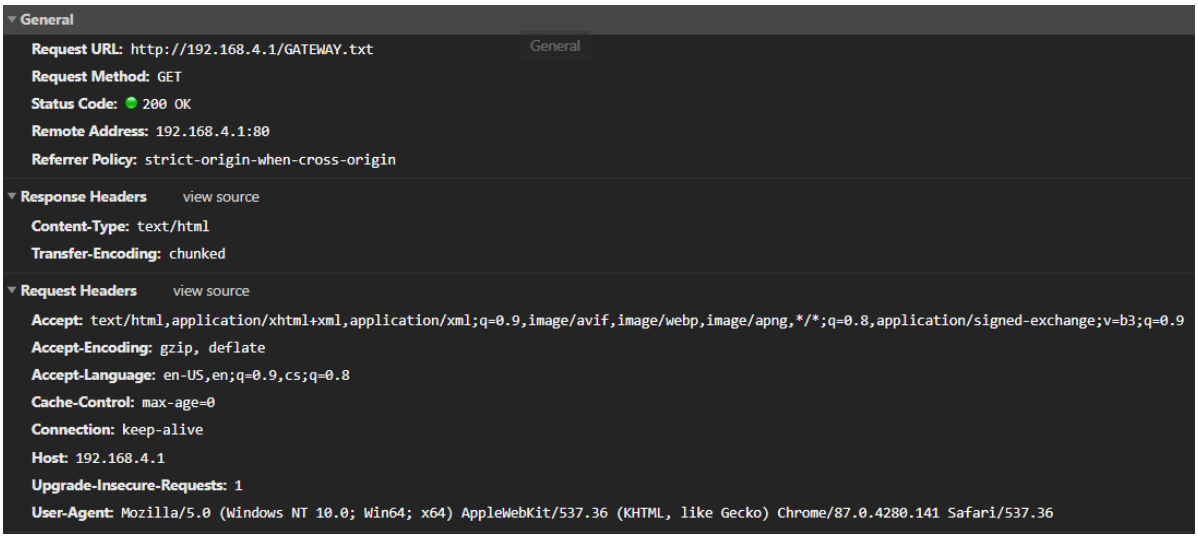
Webový server je primárně zdrojem implementované webové aplikace. Pro realizaci webového serveru bylo využito knihovny <esp_http_server.h>. Webový server dále implementuje rozhraní REST API, které slouží pro výměnu dat mezi webovou aplikací a samotným zařízením. Pro funkci sloužící ke konfiguraci a spuštění webového serveru je předdefinováno pole udávající velikost bufferu pro případ kdy se na web posílají větší soubory (například obrázky), které není možné poslat najednou. Pro toto pole je nutné před zahájením práce alokovat paměť. Následně je vytvořena výchozí HTTP konfigurační struktura obsahující port serveru (zvolen 80), maximální časy pro odeslání a příjem zpráv, atd. Poté je webový server spuštěn. REST API využívá HTTP pro propojení mezi zařízeními k odesílání, čtení a mazání dat. Uvnitř funkce pro start webového serveru jsou zaregistrovány handlery pro všechny zdroje REST API, které mohou být volány. Jedná se tedy například o URI definující požadavky jako je přihlášení uživatele, odhlášení uživatele, zapnutí sběrnicové brány, získání informací o HW, atd. viz (Kód 13.1)

```
/api/auth/sign-in
/api/auth/sign-out
/api/info
/api/function/StartStopCANfunctiononality
/api/ResetESP32
```

Listing 13.1: URI REST API

13.1 HTTP požadavek pro získání souborů tvořících web

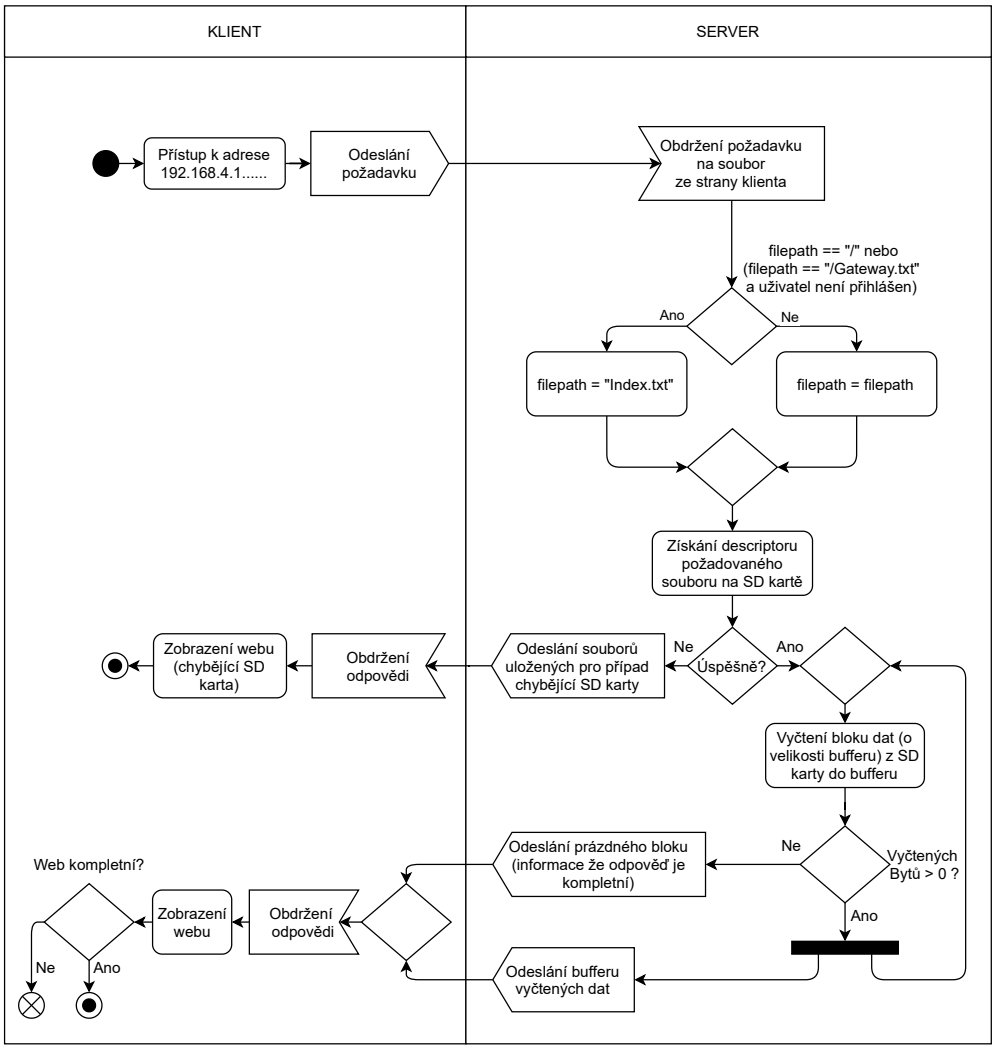
Po připojení k přístupovému bodu platformy ESP32 a pokusu o načtení webu je odeslán ze zařízení (klienta) požadavek na server. Informace o tomto požadavku lze rozdělit do tří částí viz (Obrázek 13.1).



Obrázek 13.1: HTTP metoda GET pro získání souboru ze serveru

Všeobecná hlavička obsahuje URL adresu souboru o který je žádáno. Dále obsahuje informaci o jakou HTTP metodu se jedná (v tomto případě metoda GET). Stavový kód *"200 OK"* znamená že vše proběhlo v pořádku (základní stavové kódy viz kapitola 4.2.2). Následuje adresa i s číslem portu (v tomto případě port 80 zvolený při tvorbě webového serveru). Poslední ve všeobecné části je informace o zabezpečení. Tato informace udává jakým způsobem musí být zacházeno s informacemi, které tento požadavek obsahuje.
V hlavičce odpovědi se nachází typ obsahu o který je žádáno. V tomto případě se jedná o typ *text/html* (typů je však mnohem více jako například *text/css*, *application/javascript*, *image/svg+xml*, atd.) a kódování použité pro přenos samotných požadovaných informací.
Hlavička požadavku udává kterým typům obsahu je klient schopen porozumět, jaké kódování (kompresi) obsahu je klient schopen pochopit a jakým jazykům je klient schopen porozumět (Angličtina, Čeština, atd.). Dále se zde nachází pokyny pro ukládání do mezipaměti a informace zdali má být síťové připojení zachováno i po dokončení aktuální akce (při nastavení keep-alive je propojení trvalé a není uzavřeno, což umožňuje následně provádět další požadavky na stejný server). Dalším z prvků je adresa a port na který je požadavek odesílán (pokud není zadán žádný port, pro HTTP je automaticky přiřazen port 80). Prvek Upgrade-Insecure-Requests vyjadřuje preference klienta pro šifrovanou a ověřenou odpověď. Poslední částí hlavičky požadavku je informace, jaký používá uživatel operační systém a prohlížeč.

Při samotném zpracování HTTP požadavku na soubor je zkontrolováno jestli klient žádá o úvodní (přihlašovací) stránku. Pokud žádá o stránku pro kterou je požadováno přihlášení a uživatel stále není přihlášen, je vždy automaticky přesměrován na přihlašovací stránku. Při přesměrování na úvodní stránku dochází k požadavku na HTML soubor této stránky. Následně je získán descriptor požadovaného souboru (úvodní stránky) uložené na paměťové kartě. Poté je požadavku přidělen Content-Type (v tomto případě je přidělen Content-Type *"text/html"*, protože se žádá o HTML úvodní stránky). Data jsou následně postupně čtena z SD karty (vždy je vyčteno maximálně tolik dat jaká je velikost nastaveného bufferu), data jsou poté odeslána jako HTTP odpověď na požadavek. Je zde použito Chunked Transfer Encoding a opověď je odesílána po blocích. Každému bloku předchází jeho velikost v bajtech. Tato činost je opakována do doby než je počet vyčtených bajtů roven nule. Po načtení všech požadovaných souborů je načtena kompletní úvodní stránka (Obrázek 13.2).

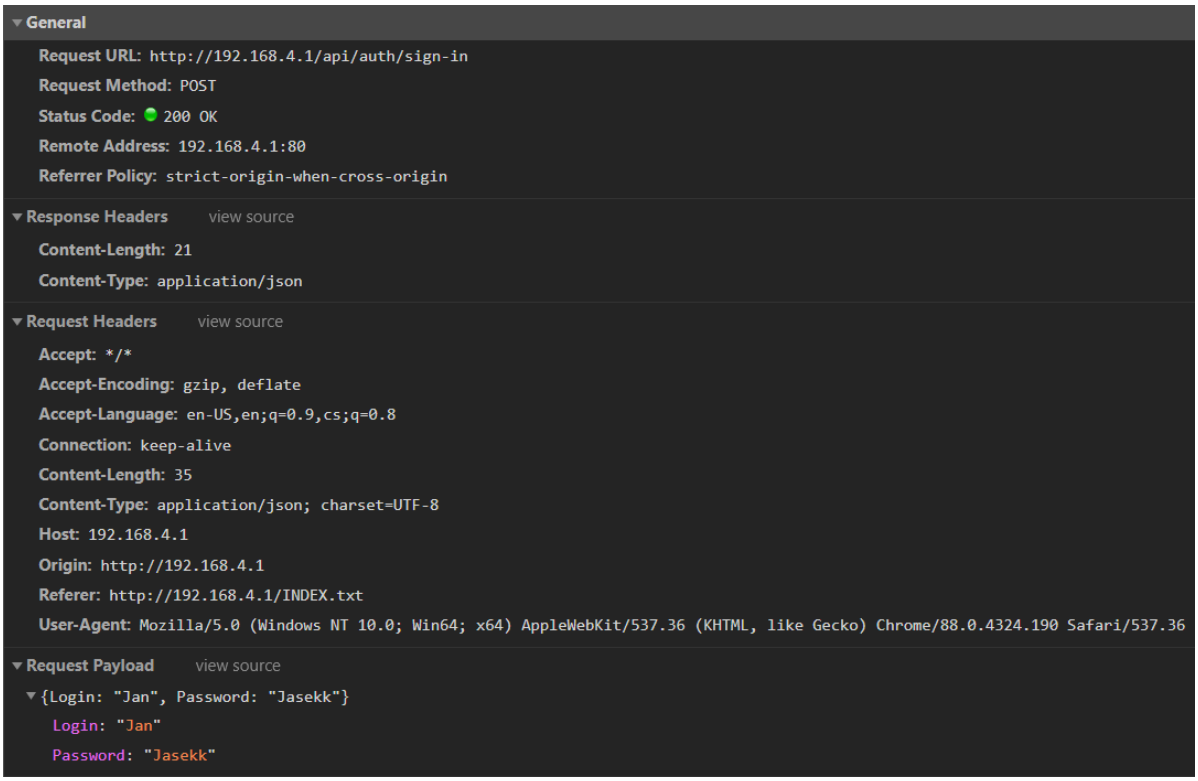


Obrázek 13.2: Aktivitní diagram komunikace klient - server při získávání souborů pro tvorbu webu

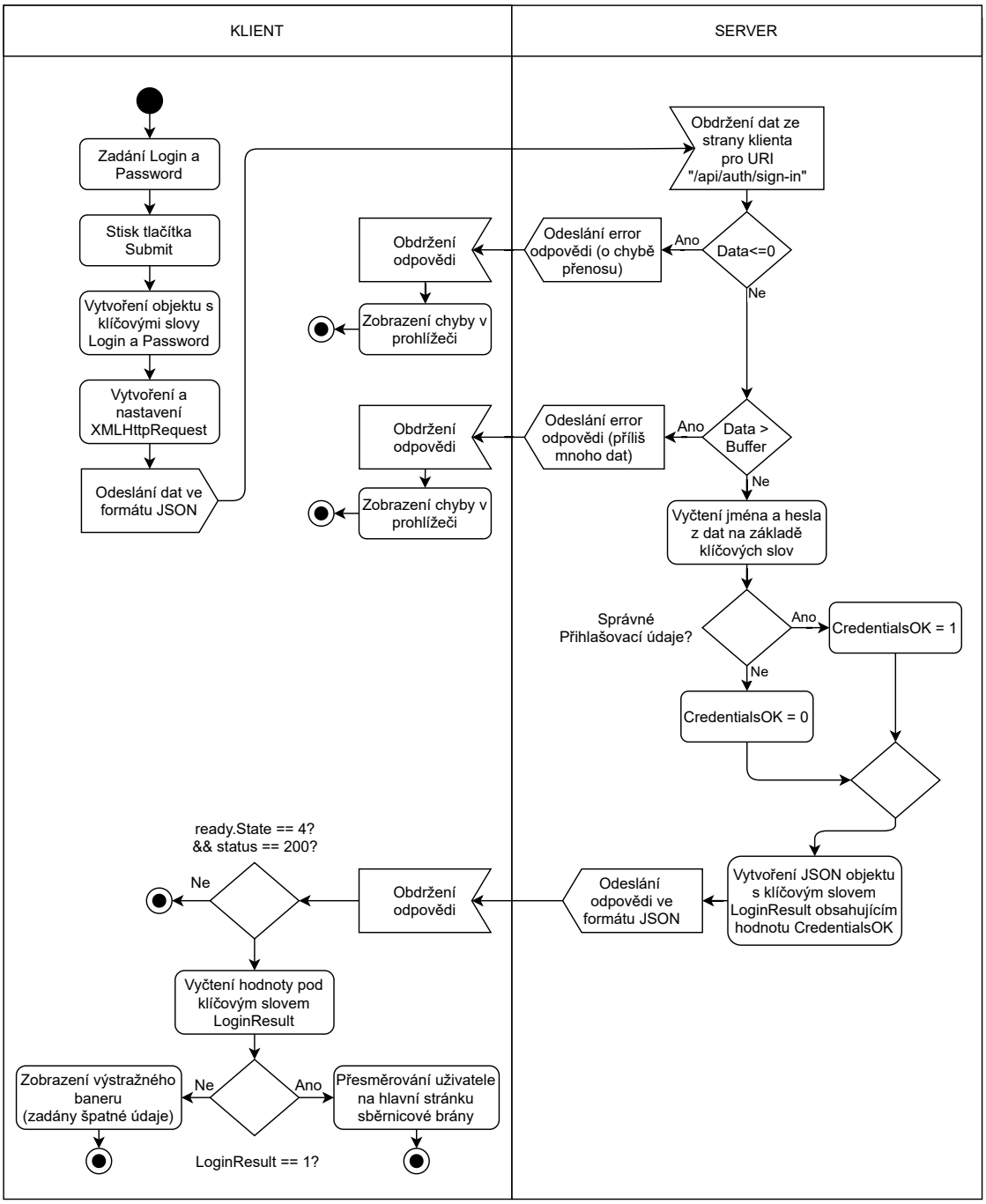
13.2 HTTP požadavek pro přenos dat ve formátu JSON

Přenos dat mezi webovou aplikací a webovým serverem je realizován pomocí HTTP. Webový server není schopen samotný požadavek zpracovat, protože nerozumí datům obsaženým v požadavku. Webový server se tedy stará o samotné provedení metody (například POST) viz (Obrázek 13.3) a následně je dle požadavku na konkrétní zdroj REST API provedeno zpracování dat pomocí kódu v programovacím jazyce C. Rozdíl mezi informacemi o požadavku při odeslání dat pomocí metody POST oproti požadavku pro získání souborů tvořících web je pouze v prvcích Request Method (která je v tomto případě POST), velikost obsahu požadavku i odpovědi a Content-Type (zde *application/json*, protože je zde použit datový formát JSON). Origin označuje původ požadavku (nezahrnuje však žádné informace o cestě). Referer umožňuje serveru identifikovat odkud konkrétně požadavek pochází (obsahuje i cestu).
Tento požadavek na přenos dat viz (Obrázek 13.3), je inicován funkcí *onPressSubmit*, která je vyvolána po stisku tlačítka *Submit* na úvodní obrazovce. Po stisku je pomocí odpovídajícího selektoru získán obsah formuláře (Login + Password). Následně je vytvořen objekt s dvěma prvky označenými klíčovými slovy "Login"a "Password". Prvku Login je přiřazen obsah uložený v proměnné Login a prvku Password je přiřazen obsah uložený v proměnné Password. Poté je vytvořen objekt XMLHttpRequest (XHR) a provedena metoda *Open* která inicializuje nově vytvořený požadavek XHR. Parametry pro metodu Open jsou požadovaná HTTP metoda (v tomto případě POST), URL adresa na kterou se má požadavek odeslat a posledním parametrem je bool parametr označující zda se má operace provést asynchronně. Dále je nastavena hlavička požadavku (tuto metodu lze zavolat několikrát s totožnou hlavičkou a hodnoty jsou posléze sloučeny do jedné hlavičky požadavku). Nastavený Content-Type je *application/json* se znakovou sadou UTF-8. Poslední částí před odesláním je převedení objektu JavaScriptu na řetězec JSON a poté je provedeno odeslání.
Požadavek je serverem zpracován a podle specifického URI požadavku je provedena funkce přiřazená tomuto URI. V této funkci je vždy zkontrolováno jestli množství příchozích dat není větší než nastavená velikost bufferu a také jestli příchozí data nejsou prázdná. Následně jsou vyčtena všechna data z HTTP požadavku. Po získání všech dat, která jsou ve formátu JSON se tento blok pomocí funkce *Parse* převede na objekt, ze kterého je již možné vyčítat informace dle klíčových slov. V požadavku pro přihlášení bylo odesláno uživatelské jméno a heslo. Toto jméno a heslo je tedy pomocí klíčových slov vyčteno z formátu JSON a následně je porovnáno se seznamem uživatelských jmen a hesel, která mají povolen přístup. V závislosti na tom zdali jsou přihlašovací údaje správné nebo nesprávné je vracena hodnota CredentialsOK = 1 nebo CredentialsOK = 0. Tato informace je pod klíčovým slovem "LoginResult" převedena do formátu JSON a odeslána jako odpověď na původní požadavek.
V okamžiku kdy je požadavek dokončen ať úspěšně (po načtení) nebo neúspěšně (po přerušení nebo chybě) je provedena kontrola vrácených hodnot readyState a status. Za předpokladu, že vše proběhlo korektně je příchozí zpráva převedena z formátu JSON na JavaScript objekt. Z vytvořeného

objektu je poté pomocí klíčového slova "LoginResult" získána hodnota označující zda uživatelem zadané jméno a heslo bylo správné či nikoli a na základě této odpovědi je uživatel buď přesměrován na hlavní stránku sběrnicové brány, nebo je ponechán na přihlašovací stránce a je upozorněn výstražným banerem, že zadané jméno nebo heslo je nesprávné.



Obrázek 13.3: HTTP metoda POST data



Obrázek 13.4: Aktivnitní diagram komunikace klient - server při přihlášení

Kapitola 14

Implementace komunikace mezi konfigurační a sběrnicovou částí sběrnicové brány

Ve formátu JSON jsou přenesena veškerá požadovaná data z webové aplikace do obvodu ESP32 viz kapitola 13.2. Sběrnicová brána (konfigurační i sběrnicová část) je realizována na jednom zařízení a není tedy nutné přenášet data pomocí dalšího komunikačního rozhraní. Data jsou mezi konfigurační a sběrnicovou částí předávána prostřednictvím globální struktury. Data jsou přenesena z webu ve tvaru ve kterém jsou na webu zadána. Volby PASS/STOP jsou přenášeny jako stringy "PASS" nebo "STOP" a pomocí programu následně upravovány do požadované podoby pro globální strukturu viz (Obrázek 14.5). Jiné volby jako například aktivní/neaktivní filtr nebo volba vstupního a výstupního kanálu pro funkcionalitu přichází z webu jako číselná hodnota, která je přímo uložena do struktury. Komunikační rychlost jednotlivých kanálů je ve struktuře zaznamenána pomocí výčtového typu s rozsahem 0 - 3, odpovídajícímu hodnotám 125, 250, 500 a 1000 kbit/s. Zda jsou jednotlivé kanály povoleny nebo ne je ve struktuře definováno stavy 0 - neaktivní, 1 - aktivní a stejně tak je definováno zda je na jednotlivých kanálech nějaká aktivita viz (Obrázek 14.1).

```
BaudrateCH0 = 0
BaudrateCH1 = 1
BaudrateCH2 = 2
EnableDisableCH0 = 1
EnableDisableCH1 = 1
EnableDisableCH2 = 1
BusActivityCH0 = 1
BusActivityCH1 = 1
BusActivityCH2 = 1
```

Obrázek 14.1: Informace o nastavení kanálů

U filtrů je předávána informace o stavu filtru 0 - neaktivní, 1 - aktivní. Parametry určující zda se jedná o filtr typu PASS nebo STOP, zda se jedná o filtr typu ID nebo DLC a zda je zadán v nějakém rozsahu ID nebo DLC nebo platí pouze pro jednu hodnotu ID nebo DLC jsou řešeny pomocí výčtových typů viz (Obrázek 14.5). Hodnota ID/DLC (ať už se jedná o jednu hodnotu, nebo je

zadán rozsah ve tvaru min;max) je poté uložena do struktury jako číslo v rozsahu 0 - 2047 nebo 0 - 8 viz (Obrázek 14.2).

Filter	enable,	Pass/Stop,	ID/DLC,	0(number)/1(range),	Condition(number),	Condition(Min),	Condition(Max)
Filter0 =	1,	0,	0,	0,	34,	0,	0
Filter1 =	1,	1,	1,	0,	4,	0,	0
Filter2 =	1,	0,	0,	1,	0,	220,	250
Filter3 =	0,	5,	5,	0,	0,	0,	0
Filter4 =	0,	5,	5,	0,	0,	0,	0
Filter5 =	0,	5,	5,	0,	0,	0,	0
Filter6 =	0,	5,	5,	0,	0,	0,	0
Filter7 =	0,	5,	5,	0,	0,	0,	0
Filter8 =	0,	5,	5,	0,	0,	0,	0
Filter9 =	0,	5,	5,	0,	0,	0,	0

Obrázek 14.2: Informace o nastavení jednotlivých filtrů

U akcí je předávána informace o stavu akce 0 - neaktivní, 1 - aktivní. Akci replaceID odpovídá hodnota 0 a samotná hodnota ID na kterou bude aktuální ID změněno (Replace to) je uložena do struktury jako číslo v rozsahu 0 - 2047 viz (Obrázek 14.3).

Action	enable,	ReplaceID/,	Replace to
Action0 =	1,	0,	290
Action1 =	1,	0,	300
Action2 =	0,	5,	0
Action3 =	0,	5,	0
Action4 =	0,	5,	0
Action5 =	0,	5,	0
Action6 =	0,	5,	0
Action7 =	0,	5,	0
Action8 =	0,	5,	0
Action9 =	0,	5,	0

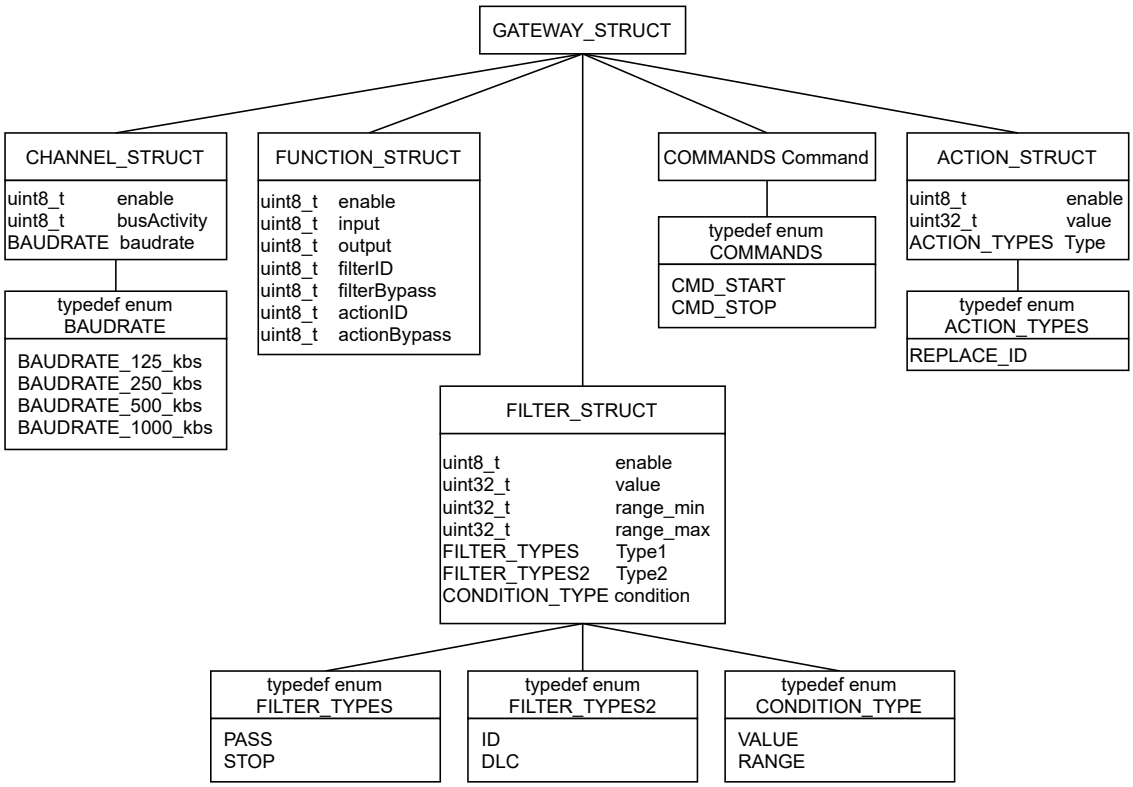
Obrázek 14.3: Informace o nastavení jednotlivých akcí

U funkcionalit je označení stavu totožné jako u filtru nebo u akce (0 - neaktivní, 1 - aktivní). ChannelIN může nabývat hodnot 0 - 2, stejně tak ChannelOUT a tyto hodnoty odpovídají zvolenému kanálu. FilterID i ActionID mohou nabývat ve funkcionalitě hodnot 0 - 9, protože může být zvoleno až 10 filtrů, ale také mohou nabývat hodnoty 100. A to v případě volby stavu Bypass. Stav Bypass je ve struktuře zaznamenáván i samostatně a to jako proměnná actionBypass a filterBypass. U obou těchto proměnných je stav 0 - neaktivní, 1 - aktivní viz (Obrázek 14.4).

Functionality	enable,	ChannelIN,	ChannelOUT,	ActionBypassed?,	ActionID,	FilterBypassed?,	FilterID
Functionality0 =	1,	0,	1,	0,	0,	0,	2
Functionality1 =	1,	2,	1,	0,	1,	0,	0
Functionality2 =	0,	0,	0,	0,	0,	0,	0
Functionality3 =	0,	0,	0,	0,	0,	0,	0
Functionality4 =	0,	0,	0,	0,	0,	0,	0
Functionality5 =	0,	0,	0,	0,	0,	0,	0
Functionality6 =	0,	0,	0,	0,	0,	0,	0
Functionality7 =	0,	0,	0,	0,	0,	0,	0
Functionality8 =	0,	0,	0,	0,	0,	0,	0
Functionality9 =	0,	0,	0,	0,	0,	0,	0

Obrázek 14.4: Informace o nastavení jednotlivých funkcionalit

Všechna tato data jsou následně uložena do struktury definované pro předávání dat mezi konfigurační a sběrnicovou částí sběrnicové brány viz (Obrázek 14.5).

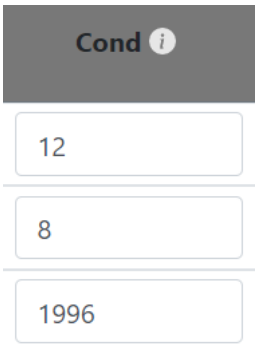


Obrázek 14.5: Struktura pro předávání dat mezi konfigurační a řídicí částí sběrnicové brány.

Kapitola 15

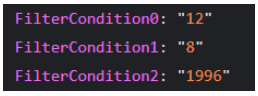
Ověření funkčnosti

V průběhu realizace práce bylo připojení k WiFi přístupovému bodu, získání souborů z SD karty, přihlašování k webu, nastavování parametrů sběrnicové brány na webu a komunikace mezi webem a webovým serverem provedeno mnohokrát. Žádná technická chyba, která by znemožnila zařízení používat nebyla od posledních změn doposud zaznamenána. V žádném z pokusů nebyla data přenášená mezi webem a webovým serverem žádným způsobem poškozena což bylo kontrolováno na několika různých úrovních viz (Obrázky 15.1, 15.2, 15.3, 15.4).

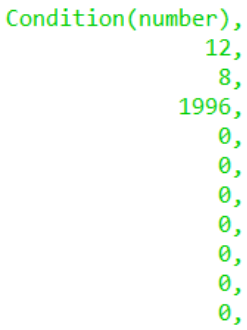


Obrázek 15.1: Uživatelem zadané hodnoty ID a DLC, které mají být filtrovány.

Obrázek 15.2: Kontrola dat na úrovni JS.



Obrázek 15.3: Kontrola dat na úrovni HTTP požadavku (zda jsou data ve formátu JSON správně uložena do Request Payloadu).



Obrázek 15.4: Kontrola dat v konzoli ESP32.

Byla zde taktéž snaha eliminovat chyby lidského faktoru. Proto jsou zde ošetřeny situace jako nevložená SD karta, špatně zadané ID, zapomenutá volba filtru, atd. Všechna tato ošetření také plní svou funkci správně.

Kapitola 16

Závěr

Tato diplomová práce byla implementována na vestavěném zařízení bez operačního systému a jejím cílem bylo realizovat konfigurační modul sběrnicové brány, což se podařilo v plné míře. Sběrnicovou bránu je tedy možno plnohodnotně konfigurovat z téměř jakéhokoli zařízení které disponuje rozhraním WiFi a internetovým prohlížečem, bez ohledu na jeho operační systém nebo velikost a rozlišení obrazovky. A to prostřednictvím webové aplikace běžící na vestavěném webovém serveru, který je realizován na HW platformě ESP32.

Prvním krokem byla volba vhodné HW platformy pro uskutečnění této práce. Jako nejoptimálnější byla shledána platforma ESP32. Tato platforma byla zvolena především z důvodu integrované WiFi a velké SW podpory. Ostatní zvažované platformy byly vyřazeny zejména z důvodu absence integrované WiFi. Dalším z důvodů vyřazení byl u některých platforem nízký výpočetní výkon, jiné z platforem byly naopak až příliš výkonné, komplikované, drahé a jejich potenciál by zůstal z naprosté části nevyužit.

Pro realizaci této práce bylo jako vývojové prostředí pro tuto platformu zvoleno prostředí Eclipse v kombinaci s ESP-IDF frameworkem. Tato kombinace byla upřednostněna před programovacím prostředím Arduino IDE z důvodu možnosti vytvářet aplikace na nižší úrovni s využitím ESP-IDF což je oficiální framework výrobce platformy ESP32.

WiFi na tomto zařízení je využita v režimu přístupového bodu a tento přístupový bod je dostupný od okamžiku, kdy je zařízení připojeno k napájení. Uživatel se k tomuto zařízení připojí jako k jakékoli jiné dostupné WiFi síti. Při zpětném pohledu by pro možné budoucí využití například i v oblasti automatizovaného testování bylo vhodnějším řešením aby nebylo zařízení přístupovým bodem, ale bylo připojeno do lokální sítě.

Během dalšího kroku, jímž byla realizace první verze webové aplikace nastal problém s nedostatečnou velikostí vnitřní paměti mikrokontroléru ESP32 pro uložení všech souborů webu. Z tohoto důvodu bylo zařízení doplněno externím modulem čtečky mikro SD karet. Na mikro SD kartě jsou uloženy všechny soubory potřebné pro tvorbu webu a současně je tak zajištěn dostatek paměti pro případné rozšiřování webové aplikace. Pro případ že by SD karta nebyla vložena je ve vnitřní paměti

zařízení uložen malý web, který je v této situaci vyvolán a informuje uživatele o problému s SD kartou.

Po tomto kroku následovala realizace webového serveru, který slouží primárně jako zdroj implementované webové aplikace. Webový server však dále implementuje rozhraní REST API s formátem JSON, které slouží pro výměnu dat mezi webovou aplikací a samotným zařízením. REST API nemusí sloužit pouze pro předávání dat s webovou aplikací jako je tomu zde, ale do budoucna je umožněna komunikace jedním komunikačním kanálem i s jinými aplikacemi. Ať už se jedná o aplikace pro iOS, Android nebo PC. Cílem práce bylo vyhnout se realizaci těchto aplikací (určených pro konkrétní OS) pro ovládání zařízení uživatelem, to však nemusí platit pro oblast automatizovaného testování, kde může být toto zařízení využito pro manipulaci se zprávami na sběrnicích v průběhu testu a ovládáno právě přes toto API z testovacího SW.

Při realizaci webové aplikace bylo využito Bootstrap front-end frameworku. Důvodem použití tohoto frameworku bylo, že se jedná o velmi populární a rozšířenou technologii pro realizaci respozivních webů, která umožňuje vytvořit webovou aplikaci komfortně použitelnou na obrazovce PC a stejně tak na displeji mobilního telefonu. Samotný web je tvořen dvěma základními částmi. První částí je přihlašovací stránka s formulářem. Druhou částí je hlavní stránka sběrnicové brány pro konfiguraci parametrů. Cílem při návrhu bylo aby stránka působila co nejvíce přehledně a co nejméně měnila své rozložení i za situace, kdy bude zobrazena na malé obrazovce a bude s ní manipulováno pomocí Bootstrap frameworku. Z tohoto důvodu byla hlavní stránka sběrnicové brány rozdělena do sekcí nastavení sběrnic, nastavení filtrů, nastavení akcí a sestavení funkcí. Stejně tak je odděleno a graficky odlišeno záhlaví a zápatí od části ve které se provádí nastavení parametrů sběrnicové brány.

Parametry zadané ve webové aplikaci jsou odesílány z webu na webový server ve formátu JSON. Zprávy jsou v zařízení zpracovány do požadované podoby a umístěny do globální struktury, která je pojítkem mezi konfigurační a sběrnicovou částí sběrnicové brány.

Snahou při realizaci této práce bylo mimo vyřešení samotného zadaného problému také řešit zadaný problém s ohledem na možnost budoucího rozšíření o další funkcionality nebo další automobilové sběrnice a využít jiných technolgií než se obvykle v této oblasti používají.

Literatura

[1] ELGERED, Johan a Jesper JANSSON. AUTOSAR Communication Stack Implementation With FlexRay [online]. In: CHALMERS UNIVERSITY OF TECHNOLOGY UNIVERSITY OF GOTHENBURG DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING. Sweden, 2012 [cit. 2020-8-17]. Dostupné z: http://publications.lib.chalmers.se/records/fulltext/156578.pdf

[2] VOSS, Wilifried. A comprehensible guide to controller area network. 2nd edition. Greenfield: Copperhill Media Corporation, [2005], 164 s. ISBN 09-765-1160-6.

[3] VBOX Automotive [online]. Buckingham: RACELOGIC UK, 1992 [cit. 2020-12-18]. Dostupné z: https://www.vboxautomotive.co.uk/index.php/en/

[4] RACELOGIC support [online]. Buckingham: RACELOGIC UK, 2020 [cit. 2020-12-18]. Dostupné z: https://racelogic.support/02VBOX_Motorsport/Modules_and_Accessories/CAN_Gateway/CAN_Gateway_User_Guide/05_-_CAN_Gateway_Setup_Software_Overview

[5] PC-3 High-performance routing and manipulation of signals in real-time. TTTech Auto [online]. Vídeň: TTTech, c2018 - 2020 [cit. 2020-12-19]. Dostupné z: https://www.tttech-auto.com/wp-content/uploads/TTTech-Automotive_PC-3-Flyer_e.pdf

[6] Gateway system for CAN and FlexRay in automotive ECU networks. IEEE Xplore [online]. New York: IEEE, 2010 [cit. 2020-12-19]. Dostupné z: https://ieeexplore.ieee.org/document/5636782

[7] PARET, Dominique. Multiplexed Networks for Embedded Systems: CAN, LIN, FlexRay, Safe-by-Wire. Society of Automotive Engineers, 2014. ISBN 978-0768019384.

[8] [Standard data frame]. In: Vector e-learning [online]. Stuttgart, Německo: Vector Informatik, c2010-2020 [cit. 2020-07-16]. Dostupné z: https://elearning.vector.com/mod/page/view.php?id=344

[9] CAN_E: CAN Bus Levels. Vector e-learning [online]. Stuttgart, Německo: Vector Informatik, c2010-2020, 2018 [cit. 2020-07-27]. Dostupné z: https://elearning.vector.com/mod/page/view.php?id=341

[10] CAN_E: CAN Controller. Vector e-learning [online]. Stuttgart, Německo: Vector Informatik, c2010-2020, 2018 [cit. 2020-07-27]. Dostupné z: https://elearning.vector.com/mod/page/view.php?id=338

[11] CAN_E: Learning Module CAN. Vector e-learning [online]. Stuttgart, Německo: Vector Informatik, c2010-2020 [cit. 2020-07-27]. Dostupné z: https://elearning.vector.com/mod/page/view.php?id=333

[12] LIN_E: Learning Module LIN. Vector e-learning [online]. Stuttgart, Německo: Vector Informatik, c2010-2020 [cit. 2020-08-17]. Dostupné z: https://elearning.vector.com/mod/page/view.php?id=309

[13] FlexRay_E: Learning Module FlexRay. Vector e-learning [online]. Stuttgart, Německo: Vector Informatik, c2010-2020 [cit. 2020-08-17]. Dostupné z: https://elearning.vector.com/mod/page/view.php?id=371

[14] YEAGER, Nancy a Robert MCGRATH. Web Server Technology. Morgan Kaufmann, 1996. ISBN 155860376X.

[15] What is a web server? - Learn web development | MDN. MDN Web Docs [online]. Mountain View, Kalifornie, USA: Mozilla Corporation, 2020, 22.5.2020 [cit. 2020-07-27]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server

[16] HTTP request methods. Resources for developers, by developers. [online]. Mountain View, USA: MDN Web Docs, Mozilla Foundation, Dec 4, 2020 [cit. 2020-12-19]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

[17] KONEČNÝ, Jaromír. Mobilní informační technologie pro řízení: Přednáška 5 - Komunikace 1. Ostrava: Technická univerzita Ostrava, 2020.

[18] ÖZLÜ, Ahmet. Mastering REST Architecture — REST Architecture Details. Medium [online]. 2018 [cit. 2021-02-01]. Dostupné z: https://ahmetozlu93.medium.com/mastering-rest-architecture-rest-architecture-details-e47ec659f6bc

[19] RS [online]. Varšava, Polsko: RS Components Sp. z o.o., 2010 [cit. 2020-10-30]. Dostupné z: https://cz.rs-online.com/web/

[20] Microchip [online]. Arizona: Microchip Technology, 2020 [cit. 2020-08-17]. Dostupné z: https://www.microchip.com/

[21] NXP Semiconductors | Automotive, Security, IoT [online]. Eindhoven: NXP Semiconductors, 2020 [cit. 2020-08-17]. Dostupné z: https://www.nxp.com/

[22] STMicroelectronics [online]. Ženeva: STMicroelectronics, 2020 [cit. 2020-08-18]. Dostupné z: https://www.st.com/content/st_com/en.html

[23] ESP-IDF Programming Guide [online]. Shanghai: Espressif Systems, 2016 - 2020 [cit. 2020-11-26]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html#

[24] Internet of Things with ESP32 [online]. [cit. 2020-11-29]. Dostupné z: http://esp32.net/

[25] Arduino [online]. 2020 [cit. 2020-11-29]. Dostupné z: https://www.arduino.cc/

[26] Server. Wikipedie [online]. San Francisco: Wikipedie, 18. 12. 2020n. l. [cit. 2020-12-19]. Dostupné z: https://cs.wikipedia.org/wiki/Server

[27] Embedded Web Servers and Application Servers: What's the Difference? [online]. North Carolina: Dzone, 2017 [cit. 2020-12-01]. Dostupné z: https://dzone.com/articles/embedded-web-servers-and-application-servers-whats

[28] Modul čtečka Micro SD karet - SPI modul. HADEX [online]. Ostrava: Softima, c2011-2021 [cit. 2021-02-01]. Dostupné z: https://www.hadex.cz/m533-modul-ctecka-micro-sd-karet---spi-modul/?gclid=Cj0KCQiA6t6ABhDMARIsAONIYyzby8alz0ECC6pjxbajUyL70Zm8J-Oae6w7XLTMlwet2PBkvQT-Ve4aAmS2EALw_wcBModulčtečkaMicroSDkaret-SPImodul.HADEX[online].Ostrava:Softima,c2011-2021[cit.2021-02-01].Dostupnéz:https://www.hadex.cz/m533-modul-ctecka-micro-sd-karet---spi-modul/?gclid=Cj0KCQiA6t6ABhDMARIsAONIYyzby8alz0ECC6pjxbajUyL70Zm8J-Oae6w7XLTMlwet2PBkvQT-Ve4aAmS2EALw_wcB

[29] Bootstrap Grid System. W3schools [online]. Refsnes Data, c1999-2021 [cit. 2021-02-23]. Dostupné z: https://www.w3schools.com/bootstrap/bootstrap_grid_system.asp

[30] HTML Introduction [online]. W3Schools, c1999-2021 [cit. 2021-02-24]. Dostupné z: https://www.w3schools.com/html/html_intro.asp

[31] CSS Introduction [online]. W3Schools, c1999-2021 [cit. 2021-02-24]. Dostupné z: https://www.w3schools.com/css/css_intro.asp

[32] JavaScript pro začátečníky: co to je a jak funguje. Tvorba webů, eshopů a aplikací [online]. Praha: Rascasone, c2021 [cit. 2021-02-24]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-javascript-pro-zacatecniky

[33] Grid system. Build fast, responsive sites with Bootstrap [online]. Bootstrap Core Team [cit. 2021-02-23]. Dostupné z: https://getbootstrap.com/docs/4.0/layout/grid/

[34] BERNERS-LEE, T., R. FIELDING a L. MASINTER. [online]. The Internet Society, 2005 [cit. 2021-03-15]. Dostupné z: https://tools.ietf.org/html/rfc3986

[35] MIESSLER, Daniel. What's the Difference Between a URI and a URL? [online]. In: . 2021 [cit. 2021-3-16]. Dostupné z: https://danielmiessler.com/study/difference-between-uri-url/#:~:text=The%20terms%20%E2%80%9CURI%E2%80%9D%20and%20%E2%80%9C,as%20HTTPs%20%2C%20FTP%20%2C%20etc.

[36] Výběr moderních webových technologií. Itnetwork [online]. c2021 [cit. 2021-03-17]. Dostupné z: https://www.itnetwork.cz/html-css/css3/zdrojakoviste/webove-aplikace/technologie-pro-vyvoj-webovych-aplikaci-vyber-technologii-2

[37] MALÝ, Martin. REST: architektura pro webové API. Zdroják.cz [online]. 3.8.2009 [cit. 2021-04-05]. Dostupné z: https://zdrojak.cz/clanky/rest-architektura-pro-webove-api/

Příloha A

Přílohy v IS Edison

- Program pro mikrokontrolér - GatewayDP.c a GatewayDP.h uloženy v souboru MCU_ESP32.zip
- Zdrojové soubory webové aplikace uložené na paměťové kartě - Všechny zdrojové soubory HTML, CSS, JS, obrázky, atd. uloženy v souboru WebSDcard.zip
- Zdrojové soubory webové aplikace uložené ve vnitřní paměti zařízení - Všechny zdrojové soubory HTML, CSS, JS, obrázky, atd. uloženy v souboru WebInternalMemory.zip